MVM

Készítette Doxygen 1.9.2

1. MVM

A Programozás Alapjai 2. - NHF *Feladatkiírás:* Tervezze meg a **Meseországi Villamos Művek (MVM)** nyilvántartási rendszerének egyszerűsített objektummodelljét, majd valósítsa azt meg! A rendszerrel minimum a következő műveleteket kívánjuk elvégezni:

- ügyfél adatinak felvétele
- szolgáltatási szerződés kötése
- szolgáltatási díj előírása (számlázás) (becsült átlagos fogyasztás)

- szolgáltatási díj befizetése
- egyenleg lekérdezése
- fogyasztás bejelentése

A rendszer lehet bővebb funkcionalitású, ezért nagyon fontos, hogy jól határozza meg az objektumokat és azok felelősségét.
Demonstrálja a működést külön modulként fordított tesztprogrammal! A megoldáshoz **ne** használjon STL tárolót!

1.1. Előrehaladás a részfeladatokban

- [x] NHF 1. - specifikáció: Pontosított feladatspecifikáció elkészítése és feltöltése PDF formátumban. A pontosított specifikáció részletesen leírja feladatot. Megadja a bemenetek és az elvárt kimenetek tartalmát és formátumát, a program működésének feltételeit, valamint rögzít minden olyan körülményt, ami egyértelműsíti a feladatot. A specifikáció a megoldásra fekete dobozként tekint, azaz a program belső felépítésével, működésével nem foglalkozik.
- [x] NHF 2. - terv: Osztálydiagram és/vagy algoritmus elkészítése és bemutatása. A részfeladat teljesítéséhez egyetlen PDF fájlba szerkesztve kell beadnia a pontosított feladatspecifikációt és a feladat megoldásához tervezett osztályok kapcsolatát és attribútumait bemutató osztálydiagramot. Feladat jellegétől függően ezt ki kell egészítenie a fontosabb algoritmusok leírásával. Az osztálydiagramot UML jelöléssel, az algoritmust tetszőleges (szöveges, folyamatábra, pszeudokód, . . .) kell megadnia.
- [x] NHF 3. - váz: Véglegesített osztályok deklarációjának illetve a főbb algoritmusok elkészítése és bemutatása. A részfeladat ellenőrzéséhez fel kell tölteni a JPorta rendszerbe olyan nem végleges tesztprogramot, ami a megtervezett osztályokat használva bemutatja azok kapcsolatát, együttműködését. A feltöltött forráskódnak szintaktikailag helyesnek kell lennie, de nem kell érdemben működnie. A JPorta rendszer csak a fordítást ellenőrzi, a futás eredményét nem, így a (tag)függvények törzse teljesen hiányozhat! E határidőt akkor tudja megfelelően teljesíteni, ha az osztályokat már olyan részletesen megtervezte, hogy a deklarációk a tervek alapján könnyen leírhatók.
- [x] NHF 4. - végleges: A feltöltött programot a feladatbeadó rendszer lefordítja és összeszerkeszti. Feltételezheti, hogy a fordítás során a CPORTA és MEMTRACE azonosítók definiáltak. A feladat csak akkor elfogadható, ha a fordítás és szerkesztés eredményeként hiba- és figyelmeztető üzenet nem keletkezik! Sikeres fordítás után a rendszer lefuttatja a programot a megadott tesztadatokkal. Sikeres elektronikus beadás után a dokumentációt és a működő programot a laborvezetőnek személyesen is be kell mutatni a tárgykövetelményben megadott határidőig.

1.2. Külső könyvtárak

- `gtest_lite`
- `memtrace`

1.3. Fogyasztás

A fogyasztás számításának képlete a következő: log2(Főbiztosíték erőssége) $*$ tarifa $*$ fogyasztás

1.4. Tesztadatok

A Teszt szövegfájlok adatai 2020. decemberét szimulálják. A következő számlázási időszak tehát 2020. 12. 31-én zárul. Eddig a dátumig a 2020. 12. 01-ig fogyasztott áram mennyiségét kell bejelenteni. Ez azt jelenti, hogy az ügyfeleknek eddig kell bejelenteniük a havi fogyasztásukat.

Az átláthatóbb és egyszerűbb tesztelés érdekében az Archvált számlák a 2020. januári számlázási időszakig nyúlnak vissza. Így a legkorábbi számlák januáriak. feltételezzük továbbá, hogy a 2019 végén minden ügyfél új mérőórát kapott, így a fogyasztási mérések 0 értékről indulnak.

1.4.1. Clientdata.txt

Az ügyfelek sample adatbázisban a következő mezők találhatóak tabulátorral elválasztva, ebben a sorrendben:

ID, **Keresztnév**, **Vezetéknév**, **Adószám**, **Város**, **Utca**, **Házszám**, (lakásszám), **telefonszám**, **e-mail cím**, **ügyfél típusa**, **születési év**, **hónap**, **nap**, **fázisok száma**, **főbiztosíték** áramerőssége (A). **kezdőegyenleg**

1.4.2. Invoices_archived.txt

Ez a fájl tartalmazza a már teljesített számlákhoz kapcsolódó adatokat, ezzel biztosítva ezen adatok perzisztenciáját.

Adatmezői a következők: **ID**(Ügyfél azonosítója), **Számla létrejöttének dátuma (Év Hónap Nap)** ,**consumption↩ Amt** (fogyasztás mennyisége), **toBePaid** (fizetendő) **Óraállás**

1.4.3. Invoices_pending.txt

Ez a fájl tartalmazza a teljesítendő számlákhoz tartozó adatokat, ezzel biztosítva ezen adatok perzisztenciáját.

Adatmezői a következők: **ID**(Ügyfél azonosítója), **Számla létrejöttének dátuma (Év Hónap Nap)** ,**consumption↩ Amt** (fogyasztás mennyisége), **toBePaid** (fizetendő) **Óraállás**

1.4.4. Consumption_announcements.txt

Ez a fájl tartalmazza a következő számlázási időszakig bejelentett fogyasztásokat. A szövegájl egy sor a következőkat tartalmazza: **Ügyfélazonosító ID bejelentés dátuma (Év Hónap Nap) Óraállás**

A mellékelt sample fájlból látható, hogy a(z) *5*, *7*, *13*,*22*, *24*, *25* számú ID-vel rendelkező Ügyfelek nem jelentették be fogyasztásukat, így ők a régebbi fogyasztásaik átlaga alapján kapnak majd számlát.

1.4.5. Tariffs.txt

Ez a fájl tarifákat tartalmaz, melyek alapján a rendszer a díjakat tudja számolni. Ezen díjak több változótól is függenek, mint pldául a fázisok számától, az ügyfél típusától, és a főbiztosíték erősségétől.

Elrendezése a következő: ∗(x jelzi azokat a cellákat, amik érvénytelenek (pl. nem létezik vállalati, 2 fázisú, 16 Amperes csomag.)∗

Ügyfél típusa / fázisok száma	16 A	32 A	63 A	128 A
Lakossági, 2 fázis			x	x
Vállalati, 2 fázis	x			
Vállalati, 3 fázis	x			

1.4.5.1. Figyelmeztetés ! A tesztadatok véletlenszerű névgenerátor oldalak segítségével, mint például a `random-name-generator.com` lettek létrehozva. Esetleges egyezés véletlen történt. Amennyiben a neve szerepel a listán és változtatást szeretne kérni, kérem, keressen fel!

! Az adószámok egy egyszerű random szám generátorral lettek elkészítve, esetleges egyezés a véletlen következtében történt.

2. Osztálymutató

2.1. Osztálylista

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

3. Fájlmutató

3.1. Fájllista

Az összes fájl listája rövid leírásokkal:

4. Osztályok dokumentációja

4.1. Address osztályreferencia

```
#include <Address.h>
```

Publikus tagfüggvények

- Address ()

- Address (String t, String str, int h, int apt=1)
- String & getStreet ()

  utca nevének lekérése
- String & getTown ()

  város nevének lekérése
- int getHouse ()

  házszám lekérése
- int getApartment ()

  Lakásszám lekérése.
- bool operator== (Address &rhs)

  Egyenlőség operátor.

Privát attribútumok

- String town
- String street

  Utcanév.
- int house
- int apartment

4.1.1. Konstruktorok és destruktorok dokumentációja

4.1.1.1. Address() [1/2] Address::Address ( ) [inline]

4.1.1.2. Address() [2/2] Address::Address (

```
String t,
String str,
int h,
int apt = 1 ) [inline]
```

4.1.2. Tagfüggvények dokumentációja

4.1.2.1. getApartment() int Address::getApartment ( )

Lakásszám lekérése.

Lakásszám lekérése

Visszatérési érték

int

4.1.2.2. getHouse() `int Address::getHouse ( )`

házszám lekérése

Házszám lekérése

Visszatérési érték

int

4.1.2.3. getStreet() `String & Address::getStreet ( )`

utca nevének lekérése

Utca nevének lekérése

Visszatérési érték

String& - String referenciájával tér vissza.

4.1.2.4. getTown() `String & Address::getTown ( )`

város nevének lekérése

Város nevének lekérése

Visszatérési érték

String& - String referenciájával tér vissza.

4.1.2.5. operator==() `bool Address::operator== ( Address & rhs )`

Egyenlőség operátor.

Egyenlőség operátor

Paraméterek

rhs	- Cím rvalue

Visszatérési érték

true , ha a két cím egyezik

false , ha a két cím nem egyezik meg.

4.1.3. Adattagok dokumentációja

4.1.3.1. apartment `int Address::apartment [private]`

Lakásszám - 1 akkor is, ha egy egylakásos házról van szó.

4.1.3.2. house `int Address::house [private]`

Házszám

4.1.3.3. street `String Address::street [private]`

Utcanév.

Figyelmeztetés

Csak az utca nevét tárolja el, típusát nem, mert nem kell a nyilvántartásba.

Így például az Almafa utca csak Almafa a rendszerben.

4.1.3.4. town `String Address::town [private]`

Város

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- Address.h
- Address.cpp

4.2. Array< T > osztálysablon-referencia

Generikus dinamikus tömb.

`#include <Array.hpp>`

Publikus tagfüggvények

- Array (size_t Siz=0)
- virtual ~Array ()
- void add (T &newElement)

  Hozzáad egy elemet a tömb végéhez.
- T & get (size_t index) const

  Visszaad egy adott indexű elemet.
- size_t size () const

  Visszaadja a tömb méretét.
- T * begin () const

  Visszad egy pointert a legelső elemre.
- T * end () const

  Visszad egy pointert az utolsó elem után.
- int isElement (T &e) const

  Megnézi, hogy egy adott elem benne van-e a tömbben.
- void del (size_t indx)

  Töröl egy adott indexűelemet.
- void del (T &e)

  Töröl egy adott elemet a tömbből.
- T & operator[ ] (size_t index) const

  Indexelő operátor egy adott indexű elem refernciájával tér vissza.
- Array & operator= (const Array &rhs)

  Értékadó operátor.

Privát attribútumok

- T * data
- size_t used

4.2.1. Részletes leírás

template<class T>
class Array< T >

Generikus dinamikus tömb.

Sablon paraméterek

T	A tömbben tárolandó objektumok típusa.

4.2.2. Konstruktorok és destruktorok dokumentációja

4.2.2.1. Array() `template<class T >`

```
Array< T >::Array (
            size_t Siz = 0 )  [inline]
```

Konstruktor. alapesetben nulla elemű.

4.2.2.2. ~Array() template<class T >

```
virtual Array< T >::~Array ( )  [inline], [virtual]
```

Destruktor

4.2.3. Tagfüggvények dokumentációja

4.2.3.1. add() template<class T >

```
void Array< T >::add (
            T & newElement )  [inline]
```

Hozzáad egy elemet a tömb végéhez.

Paraméterek

newElement	- a hozzáadandó elem referenciája.

4.2.3.2. begin() template<class T >

```
T * Array< T >::begin ( ) const  [inline]
```

Visszad egy pointert a legelső elemre.

Visszatérési érték

T∗ - pointer a legelső elemre.

4.2.3.3. del() [1/2] template<class T >

```
void Array< T >::del (
            size_t indx )  [inline]
```

Töröl egy adott indexűelemet.

indexelési hiba esetén std::out_of_range hibát dob

Paraméterek

indx	- a törlendő index

4.2.3.4. del() [2/2] template<class T >

```
void Array< T >::del (
            T & e )  [inline]
```

Töröl egy adott elemet a tömbből.

std::logic_error -t dob, ha a törölni kívánt elem nem tagja az arraynek.

Paraméterek

e	- a törlendő elem.

4.2.3.5. end() template<class T >

```
T * Array< T >::end ( ) const  [inline]
```

Visszad egy pointert az utolsó elem után.

Intervallum ok az arrayben: [begin,end[

Visszatérési érték

T∗ pointer az Array utolsó elemének utánra.

4.2.3.6. get() template<class T >

```
T & Array< T >::get (
            size_t index ) const  [inline]
```

Visszaad egy adott indexű elemet.

hasonló az operator[]-hoz.

Ha túlindexelünk, std::out_of_range kivételt dob.

Paraméterek

index	a keresendő elem idexe

Visszatérési érték

T& Az adott indexen levő objektum referenciája.

4.2.3.7. isElement() template<class T >

```
int Array< T >::isElement (
            T & e ) const  [inline]
```

Megnézi, hogy egy adott elem benne van-e a tömbben.

Paraméterek

e	- a keresendő elem

Visszatérési érték

int -1, ha nincs benne, különben az elem indexével tér vissza.

4.2.3.8. operator=() `template<class T >`

```
Array & Array< T >::operator= (
            const Array< T > & rhs )  [inline]
```

Értékadó operátor.

Paraméterek

rhs	- másolandó rvalue

Visszatérési érték

Array& - az új tömb referenciájával tér vissza.

4.2.3.9. operator[]() `template<class T >`

```
T & Array< T >::operator[] (
            size_t index ) const  [inline]
```

Indexelő operátor egy adott indexű elem refernciájával tér vissza.

indexelési hiba esetén std::out_of_range hibát dob

Paraméterek

index	- a kért index

Visszatérési érték

T& - A kért indexen levő obejtuk referenciájával tér vissza.

4.2.3.10. size() `template<class T >`

```
size_t Array< T >::size ( ) const  [inline]
```

Visszaadja a tömb méretét.

Visszatérési érték

size_t - méret

4.2.4. Adattagok dokumentációja

4.2.4.1. data `template<class T >`

`T* Array< T >::data [private]`

Adatra mutató pointer

4.2.4.2. used `template<class T >`

`size_t Array< T >::used [private]`

A tömbben levő elemek darabszáma

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- Array.hpp

4.3. Client osztályreferencia

`#include <Client.h>`

Publikus tagfüggvények

- Client ()
- Client (int id, String lN, String fN, Date b, Address res, String m, String em, String taxN, bool type, int phases, int strength)
- Client & operator= (Client &rhs)

  Értékadó operátor.
- size_t ClientSize ()
- void addFunds (double moneyVal)

  Egyenleget ír jóvá az ügyfél számláján.
- double getBalance () const

  Lekérdezi az ügyfél egyenlegét.
- int getId () const

  Lekérdzei az ügyfél azonosítóját.
- String getPhoneNumber () const

  Lekérdezi az ügyfél telefonszámát.
- String getEMail () const

  Lekérdezi az ügyfél e-mail címét.
- Date & getDate ()

  Lekérdezi az ügyfél születési dátumát.
- String getfirstName () const

  Lekérdezi az ügyfél keresztnevét.
- String getlastName () const

Lekérdezi az ügyfél vezetéknevét.

- Address & getAddress ()

  Lekérdezi az ügyfél lakcímét.

- int getPhases () const

  Lekérdezi a fázisok számát.

- int getStrength () const

  Lekérdezi a főbiztosíték áramerősségét.

- bool getType () const

  Lekérdezi az ügyfél típusát.

- String getTN () const
- int getElectricMeterVal () const
- double getDebtval () const

  Tartozás mennyiségének lekérése.

- void pay_Pending_Invoices ()

  Megpróbálja befizetni a befizetésre váró számlákat.

Publikus attribútumok

- Consumption_announcement announcement
- Array< Invoice > archivedInvoices
- Array< Invoice > pendingInvoices

Védett tagfüggvények

- void modify_electricMeter (int amt)

  Villanyóra állásának módosítása.

Privát attribútumok

- int id
- String firstName
- String lastName
- Date born
- Address resAddress
- String mobile
- String e_mail
- String taxNumber
- int electricMeter_last
- bool type
- int phases
- int strength
- double balance

Barátok

- class Controller
- std::ostream & operator<< (std::ostream &os, Client &c)

  stream operátor

4.3.1. Konstruktorok és destruktorok dokumentációja

4.3.1.1. Client() [1/2] Client::Client ( ) [inline]

4.3.1.2. Client() [2/2] Client::Client (

```
int id,
String lN,
String fN,
Date b,
Address res,
String m,
String em,
String taxN,
bool type,
int phases,
int strength ) [inline]
```

4.3.2. Tagfüggvények dokumentációja

4.3.2.1. addFunds() void Client::addFunds (

```
double moneyVal )
```

Egyenleget ír jóvá az ügyfél számláján.

Összeget ír jóvá az ügyfél számláján.

Paraméterek

moneyVal	

4.3.2.2. ClientSize() size_t Client::ClientSize ( )

4.3.2.3. getAddress() Address & Client::getAddress ( )

Lekérdezi az ügyfél lakcímét.

Lakcím lekérdezése

Visszatérési érték

Address&

4.3.2.4. getBalance() `double Client::getBalance ( ) const`

Lekérdezi az ügyfél egyenlegét.

Lekérdezi az ügyfél egyenlegét.

Visszatérési érték

double

4.3.2.5. getDate() `Date & Client::getDate ( )`

Lekérdezi az ügyfél születési dátumát.

Lekérdezi az ügyfél születési dátumát.

Visszatérési érték

Date& a dátum referenciájával tér vissza.

4.3.2.6. getDebtval() `double Client::getDebtval ( ) const`

Tartozás mennyiségének lekérése.

Tartozás mennyiségének kiszámolása.

Összegzi a befizetésre váró számlák fizetendőit.

Visszatérési érték

double

4.3.2.7. getElectricMeterVal() `int Client::getElectricMeterVal ( ) const`

Lekérdezi a villanyóra azon állását, amediig be van fizetve

Visszatérési érték

int

4.3.2.8. getEMail() `String Client::getEMail ( ) const`

Lekérdezi az ügyfél e-mail címét.

lekérdezi az ügyfél e-mail címét

Visszatérési érték

String

4.3.2.9. getfirstName() `String Client::getfirstName ( ) const`

Lekérdezi az ügyfél keresztnevét.

Lekérdezi az ügyfél keresztnevét.

Visszatérési érték

String

4.3.2.10. getId() `int Client::getId ( ) const`

Lekérdzei az ügyfél azonosítóját.

Lekérdezi az Ügyfél azonosítóját.

Visszatérési érték

int

4.3.2.11. getlastName() `String Client::getlastName ( ) const`

Lekérdezi az ügyfél vezetéknevét.

Lekérdezi az ügyfél vezetéknevét.

Visszatérési érték

String

4.3.2.12. getPhases() `int Client::getPhases ( ) const`

Lekérdezi a fázisok számát.

Fázisok számának lekérdezése

Visszatérési érték

int

4.3.2.13. getPhoneNumber() `String Client::getPhoneNumber ( ) const`

Lekérdezi az ügyfél telefonszámát.

lekérdezi az ügyfél telefonszámát.

Visszatérési érték

String

4.3.2.14. getStrength() `int Client::getStrength ( ) const`

Lekérdezi a főbiztosíték áramerősségét.

Főbiztosíték erősségének lekérdezése

Visszatérési érték

int

4.3.2.15. getTN() `String Client::getTN ( ) const`

Lekérdezi az ügyfél/vállalat adóazonosítóját.

Visszatérési érték

String

4.3.2.16. getType() `bool Client::getType ( ) const`

Lekérdezi az ügyfél típusát.

Visszaadja az ügyfél típusát (magánszemély/vállalati)

Visszatérési érték

true , ha vállalati
false , ha lakossági

4.3.2.17. modify_electricMeter() `void Client::modify_electricMeter ( int amt ) [protected]`

Villanyóra állásának módosítása.

Villanyóra állásának módosítása

Paraméterek

amt	- a mennyiség, amennyivel módosítandó.

4.3.2.18. operator=() `Client & Client::operator= ( Client & rhs )`

Értékadó operátor.

Értékadó operatáror

Paraméterek

rhs	a másolandó Client obejktum referenciája

Visszatérési érték

Client& - az új Client objektum referenciája.

4.3.2.19. pay_Pending_Invoices() `void Client::pay_Pending_Invoices ( )`

Megpróbálja befizetni a befizetésre váró számlákat.

Megpróbálja befizetni a befizetésre váró számlákat.

1. Kronologikus sorrendben halad, a legrégebbi tartozástól a legújabbig.
2. Addig halad, míg van elegendő fedezet a tartozások teljesítésére.

4.3.3. Barát és kapcsolódó függvények dokumentációja

4.3.3.1. Controller `friend class Controller [friend]`

4.3.3.2. operator<< `std::ostream & operator<< ( std::ostream & os, Client & c ) [friend]`

stream operátor

Stream operator

Paraméterek

os	- stream
c	- Ügyfél obejmtum referenciája

Visszatérési érték

std::ostream& - stream

4.3.4. Adattagok dokumentációja

4.3.4.1. announcement `Consumption_announcement Client::announcement`

Fogyasztási bejelentés

4.3.4.2. archivedInvoices `Array<Invoice> Client::archivedInvoices`

Archivált számlák

4.3.4.3. balance `double Client::balance [private]`

Ügyfél egyenlege

4.3.4.4. born `Date Client::born [private]`

Ügyfél születési dátuma

4.3.4.5. e_mail `String Client::e_mail [private]`

Ügyfél e-mail címe

4.3.4.6. electricMeter_last `int Client::electricMeter_last [private]`

Ügyfél utolsó, fizetett óraállása

4.3.4.7. firstName `String Client::firstName [private]`

Ügyfél keresztneve

4.3.4.8. id `int Client::id [private]`

Ügyfél azonosítója

4.3.4.9. lastName `String Client::lastName [private]`

Ügyfél vezetékneve

4.3.4.10. mobile `String Client::mobile [private]`

Ügyfél telefonszáma

4.3.4.11. pendingInvoices `Array<Invoice> Client::pendingInvoices`

Befizetésre váró számlák

4.3.4.12. phases `int Client::phases [private]`

Ügyfél csomagjának fázisszáma (2,3)

4.3.4.13. resAddress `Address Client::resAddress [private]`

Ügyfél címe

4.3.4.14. strength `int Client::strength [private]`

Ügyfél főbiztosítékjának erőssége.

4.3.4.15. taxNumber `String Client::taxNumber [private]`

Ügyfél adóazonoító jele

4.3.4.16. type `bool Client::type [private]`

Ügyfél típusa (lakossági / vállalati)

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- Client.h
- Client.cpp

4.4. Consumption_announcement osztályreferencia

`#include <Consumption_announcement.h>`

Publikus tagfüggvények

- Consumption_announcement ()
- Consumption_announcement (Date d, int emVal)
- Date & getDate ()

  Bejelentés dátumának lekérése.
- int get_EM_val ()

  Bejelentéskori óraállás lekérése.
- void Reset ()

  Visszaállítja a bejelentett óraállást -1 -re.
- bool operator== (Consumption_announcement &rhs)

  Egyenlőség operátor.

Privát attribútumok

- Date date
- int electricMeterVal

4.4.1. Konstruktorok és destruktorok dokumentációja

4.4.1.1. Consumption_announcement() [1/2] Consumption_announcement::Consumption_announcement ( ) [inline]

4.4.1.2. Consumption_announcement() [2/2] Consumption_announcement::Consumption_announcement (Date *d*, int *emVal*) [inline]

4.4.2. Tagfüggvények dokumentációja

4.4.2.1. get_EM_val() int Consumption_announcement::get_EM_val ( )

Bejelentéskori óraállás lekérése.

Bejelentkéskori óraállás lekérése

Visszatérési érték

int - óraállás

4.4.2.2. getDate() `Date & Consumption_announcement::getDate ( )`

Bejelentés dátumának lekérése.

Bejelentés dátumának lekérése

Visszatérési érték

Date& - dátum referenciával tér vissza.

4.4.2.3. operator==() `bool Consumption_announcement::operator== ( Consumption_announcement & rhs )`

Egyenlőség operátor.

Egyenlőség operátor

Paraméterek

rhs	- fogyasztás bejelentés rlvalue

Visszatérési érték

true , ha a két bejelentés egyezik.
false , ha a két bejelentés nem egyezik.

4.4.2.4. Reset() `void Consumption_announcement::Reset ( )`

Visszaállítja a bejelentett óraállást -1 -re.

-1 beállításával a menüből új fogyasztás jelenthető be.

4.4.3. Adattagok dokumentációja

4.4.3.1. date `Date Consumption_announcement::date [private]`

Bejelentés dátuma

4.4.3.2. electricMeterVal `int Consumption_announcement::electricMeterVal [private]`

Bejelentkéskori óraállás.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- Consumption_announcement.h
- Consumption_announcement.cpp

4.5. Controller osztályreferencia

Az MVM rendszert kezelő osztály, amely a nyilvántartást is tartalmazza.

`#include <Controller.h>`

Publikus tagfüggvények

- Controller ()
- void loadData (char const ∗CData, char const ∗Invoices, char const ∗Invoices_pending, char const ∗Tariffs, char const ∗CAnnFile)

  Az adatokat tartalmazó fájlok betöltése, tárolása.
- void saveData (char const ∗CData, char const ∗Invoices, char const ∗Invoices_pending_file, char const ∗CAnnFile)

  Az adatokat tartalmazó fájlok mentése.
- void create_Invoices (Date &todayDate)

  Számlák létrehozása, számlázási időszak lezárása.
- void newClient (Client &c)

  Új Ügyfél hozzáadása a tárolóhoz.
- void announceConsumption (Client &c, int emVal, Date &d)

  Fogyasztást jelent be.
- Client & getClient (size_t id)

  Visszaad egy ügyfél objektum referenciát, az ügyfél ID-je alapján.
- double calculate_toBePaid (Client &c)

  Fizetendő összeg kiszámolása.
- size_t clientsCount ()

  Visszaadja, hány ügyfél van a rendszerben.

Privát attribútumok

- Array< Client > clients

4.5.1. Részletes leírás

Az MVM rendszert kezelő osztály, amely a nyilvántartást is tartalmazza.

4.5.2. Konstruktorok és destruktorok dokumentációja

4.5.2.1. Controller() `Controller::Controller ( ) [inline]`

4.5.3. Tagfüggvények dokumentációja

4.5.3.1. announceConsumption()

```
void Controller::announceConsumption (
        Client & c,
        int emVal,
        Date & d )
```

Fogyasztást jelent be.

Csak akkor állítja be a fogyasztást, ha -1 volt addig -> vagyis ha még nem volt bejelentés a hónapra.

Paraméterek

c	- Ügyfél objektum refernciája.
emVal	- Az óra állása a bejelentés pillanatában (nem fogyasztás)
d	- A bejelentés dátum referenciája

4.5.3.2. calculate_toBePaid() `double Controller::calculate_toBePaid ( Client & c )`

Fizetendő összeg kiszámolása.

Olyan esetekben történik ennek a függvénynek a számolása, ha

1. - Van az adott időszakra fogyasztási bejelentés (debug: zöld) 2- - Ki lett számítva egy átlag fogyasztás, ami alapján számolhat a rendszer. (debug: sárga)

A számlák ilyenkor már elemei a tárolónka, 0 értékkel.

A Tarifa számításának módja: log2(Főbiztosíték erőssége)∗tarifa∗fogyasztás+alapdíj

Paraméterek

c	- Referencia az ügyfél objektumára, akinek éppen számlázunk.

Visszatérési érték

double A fizetendő értéke, double típusként.

4.5.3.3. clientsCount() `size_t Controller::clientsCount ( )`

Visszaadja, hány ügyfél van a rendszerben.

Visszatérési érték

size_t - a rendszerben tárolt ügyfelek száma

4.5.3.4. create_Invoices() `void Controller::create_Invoices ( Date & todayDate )`

Számlák létrehozása, számlázási időszak lezárása.

A dátum a szimulált működésben ez 2020. 12. 31., ugyanis az ügyfelek éppen a novemberi fogyasztásukat jelentik be.

Paraméterek

todayDate	A "mai" dátum.

Minden egyes kliensre meghívódik:

1. Ha van az ügyfélnek fogyasztási bejelentése az időszakra, akkor ez alapján számoljunk!
2. Ha nincs, akkor az archivált számlák alapján határozzunk meg egy átlagot, majd ennek vegyük a fogyasztását, az órájukat is az átlag szerint toljuk tovább.

Ha nem volt még archivált számlája (új ügyfél), akkor a rendszer 1kWh fogyasztást mér fel, fizetendőnek pedig 30 000 Huf-t számol fel. kell fizetnie.

4.5.3.5. getClient()

```
Client & Controller::getClient (
        size_t id )
```

Visszaad egy ügyfél objektum referenciát, az ügyfél ID-je alapján.

Paraméterek

id	A keresett ügyfél ID-je.

Visszatérési érték

Client& - az ügyfél objektum refernciája.

4.5.3.6. loadData()

```
void Controller::loadData (
        char const * CData,
        char const * Invoices,
        char const * Invoices_pending,
        char const * Tariffs,
        char const * CAnnFile )
```

Az adatokat tartalmazó fájlok betöltése, tárolása.

Paraméterek

CData	- Az ügyfelek adatait tartalmazó, tabulátoral tagolt szövegfájl.
Invoices	- Az archivált számlákat tartalmazó, tabulátoral tagolt szövegfájl.
Invoices_pending	- A befizetésre váró számlákat tartalmazó, tabulátoral tagolt szövegfájl.
Tariffs	- A tarifákat tartalmazó, tabulátoral tagolt szövegfájl.
CAnnFile	- A fogyasztási bejelentéseket tartalmazó, tabulátoral tagolt szövegfájl.

Ügyfelek adatainak betölése

Befizetett számlák betölése

Tarifák betölése

Fogyasztási bejelentések betöltése

Befizetésre váró számlák betöltése

- befizetésre váró számlát a rendszer generál.

4.5.3.7. newClient() `void Controller::newClient ( Client & c )`

Új Ügyfél hozzáadása a tárolóhoz.

Ekkor már a feltöltendő ügyfél adataiból össze lett állítva az ügyfél.

Paraméterek

c	- A feltöltenő Ügyfél objektuma.

4.5.3.8. saveData() `void Controller::saveData ( char const * CData, char const * Invoices, char const * Invoices_pending_file, char const * CAnnFile )`

Az adatokat tartalmazó fájlok mentése.

Az adatokat tartalmazó fájlok mentése, adatok tárolása.

Bezárás parancsra (7) bezárul. A Unit teszt nem használja, ,mert például a számlázás, és fizetés után az adatok megváltoznak, a következő futáskor a teszt hibásan futna le.

Használatának kipróbálásához cseréljük le a main.cpp-t az mvm_with_menu.cpp állományram használjuk a CLI verziót.

Paraméterek

CData	- Az ügyfelek adatait tartalmazó, tabulátoral tagolt szövegfájl.
Invoices	- Az archivált számlákat tartalmazó, tabulátoral tagolt szövegfájl.
Invoices_pending_file	- A befizetésre váró számlákat tartalmazó, tabulátoral tagolt szövegfájl.
CAnnFile	- A fogyasztási bejelentéseket tartalmazó, tabulátoral tagolt szövegfájl.

4.5.4. Adattagok dokumentációja

4.5.4.1. clients `Array<Client> Controller::clients [private]`

Az ügyfeleket tartalmazó tömb.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- Controller.h
- Controller.cpp

4.6. Date osztályreferencia

`#include <Date.h>`

Publikus tagfüggvények

- Date ()
- Date (unsigned int y, unsigned int m, unsigned int d)
- unsigned int getYear ()

  A dátumban tárolt év visszaadása.
- unsigned int getMonth ()

  A dátumban tárolt hónap visszaadása.
- unsigned int getDay ()

  A dátumban tárolt nap visszaadása.
- bool operator== (Date &rhs)

  Két dátum közti egyenlőséget vizsgáló operator.
- bool operator< (Date &rhs)

  Megállapítja, hogy a dátum kisebb-e az rvaluenál.
- bool operator<= (Date &rhs)
- bool operator> (Date &rhs)

  Megállapítja, hogy a dátum nagyobb-e az rvaluenál.
- bool operator>= (Date &rhs)

  Megállapítja, hogy a dátum nagyobb VAGY egyenlő-e az rvaluenál.

Privát attribútumok

- unsigned int year
- unsigned int month
- unsigned int day

Barátok

- std::ostream & operator<< (std::ostream &os, Date &d)

  Kiír az ostream-re.

4.6.1. Konstruktorok és destruktorok dokumentációja

4.6.1.1. Date() [1/2] Date::Date ( ) [inline]

4.6.1.2. Date() [2/2] Date::Date (
unsigned int *y*,
unsigned int *m*,
unsigned int *d*) [inline]

Default konstruktor

4.6.2. Tagfüggvények dokumentációja

4.6.2.1. getDay() unsigned int Date::getDay ( ) [inline]

A dátumban tárolt nap visszaadása.

Visszatérési érték

unsigned int

4.6.2.2. getMonth() unsigned int Date::getMonth ( ) [inline]

A dátumban tárolt hónap visszaadása.

Visszatérési érték

unsigned int

4.6.2.3. getYear() unsigned int Date::getYear ( ) [inline]

A dátumban tárolt év visszaadása.

Visszatérési érték

unsigned int

4.6.2.4. operator<() bool Date::operator< (
Date & *rhs*) [inline]

Megállapítja, hogy a dátum kisebb-e az rvaluenál.

Paraméterek

rhs	- Dátum típusú objektum

Visszatérési érték

true , ha lvalue kisebb, mint rvalue.

false , ha lvalue nagyobb, mint rvalue.

4.6.2.5. operator<=() `bool Date::operator<= ( Date & rhs ) [inline]`

4.6.2.6. operator==() `bool Date::operator== ( Date & rhs ) [inline]`

Két dátum közti egyenlőséget vizsgáló operator.

Paraméterek

rhs	- Dátum típusú objektum

Visszatérési érték

true , ha a két dátum egyezik

false , ha a két dátum enm egyezik.

4.6.2.7. operator>() `bool Date::operator> ( Date & rhs ) [inline]`

Megállapítja, hogy a dátum nagyobb-e az rvaluenál.

Paraméterek

rhs	- Dátum típusú objektum

Visszatérési érték

true , ha lvalue nagyobb, mint rvalue.

false , ha lvalue kisebb, mint rvalue.

4.6.2.8. operator>=() `bool Date::operator>= ( Date & rhs ) [inline]`

Megállapítja, hogy a dátum nagyobb VAGY egyenlő-e az rvaluenál.

Paraméterek

rhs	- Dátum típusú objektum

Visszatérési érték

true , ha lvalue nagyobb VAGY egyenlő az rvalueval.

false , ha lvalue kisebb, VAGY egyenlő az rvalueval.

4.6.3. Barát és kapcsolódó függvények dokumentációja

4.6.3.1. operator<< `std::ostream & operator<< ( std::ostream & os, Date & d ) [friend]`

Kiír az ostream-re.

Stream operator

Paraméterek

os	- ostream objektum
d	- kiírandó dátum.

Visszatérési érték

std::ostream&

4.6.4. Adattagok dokumentációja

4.6.4.1. day `unsigned int Date::day [private]`

A tárolandó nap

4.6.4.2. month `unsigned int Date::month [private]`

A tárolandó hónap

4.6.4.3. year `unsigned int Date::year [private]`

A tárolandó év

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- Date.h

4.7. Invoice osztályreferencia

Egy számla osztálya.

`#include <Invoice.h>`

Publikus tagfüggvények

- Invoice ()
- Invoice (Date c, Consumption_announcement &cAnnounce, int cAmt)
  Konstruktor.
- Date & getCreated ()
  létrheozás dátumának lekérése.
- Consumption_announcement & getCAnn ()
  Fogyasztási bejelentés lekérése.
- void set_toBePaid (double what)
  Fizetendő összeg beállítása.
- int getConsumptionAmt () const
  Fogyasztás mennyiségének lekérdezése.
- double get_toBePaid () const
  Fizetendő összeg lekérése.
- bool operator== (Invoice &rhs)
  Egyenlőség operátor.

Privát attribútumok

- Date created
- Consumption_announcement announcement
- int consumptionAmt
- double toBePaid

Barátok

- std::ostream & operator<< (std::ostream &os, Invoice &c)
  stream operátor

4.7.1. Részletes leírás

Egy számla osztálya.

4.7.2. Konstruktorok és destruktorok dokumentációja

4.7.2.1. Invoice() [1/2] Invoice::Invoice ( ) [inline]

4.7.2.2. Invoice() [2/2] Invoice::Invoice (Date c, Consumption_announcement & cAnnounce, int cAmt) [inline]

Konstruktor.

Paraméterek

c	A létrehozandó számla dátuma.
cAnnounce	A számlához tartozó fogyasztási bejelentés.
cAmt	A fogyasztás mennyisége

4.7.3. Tagfüggvények dokumentációja

4.7.3.1. get_toBePaid() double Invoice::get_toBePaid ( ) const

Fizetendő összeg lekérése.

Fizetendő összeg lekérdezése

Visszatérési érték

double - fizetendő összeg.

4.7.3.2. getCAnn() Consumption_announcement & Invoice::getCAnn ( )

Fogyasztási bejelentés lekérése.

Fogyasztási bejelentés lekérése

Visszatérési érték

Consumption_announcement& - fogyasztási bejelentés referenciája

4.7.3.3. getConsumptionAmt() `int Invoice::getConsumptionAmt ( ) const`

Fogyasztás mennyiségének lekérdezése.

Fogyasztás mennyiségének lekérdezése

Visszatérési érték

int - fogyasztás mennyisége

4.7.3.4. getCreated() `Date & Invoice::getCreated ( )`

létrheozás dátumának lekérése.

Létrehozás dátumának lekérdezése

Visszatérési érték

Date& - dátum referencia

4.7.3.5. operator==() `bool Invoice::operator== ( Invoice & rhs )`

Egyenlőség operátor.

Egyenlőség operator.

Paraméterek

rhs	- rval számla

Visszatérési érték

true , ha a két számla megegyezik
false , ha a két számla nem egyezik meg.

4.7.3.6. set_toBePaid() `void Invoice::set_toBePaid ( double what )`

Fizetendő összeg beállítása.

Fizetendő összeg beállítása

Paraméterek

what	- beállítandó összeg.

4.7.4. Barát és kapcsolódó függvények dokumentációja

4.7.4.1. operator<< `std::ostream & operator<< ( std::ostream & os, Invoice & c ) [friend]`

stream operátor

Stream operator

Paraméterek

os	- stream
i	- Számla

Visszatérési érték

std::ostream& - stream

4.7.5. Adattagok dokumentációja

4.7.5.1. announcement `Consumption_announcement Invoice::announcement [private]`

A számlához tartozó fogyasztási bejelentés

4.7.5.2. consumptionAmt `int Invoice::consumptionAmt [private]`

Fogyasztás mennyisége

4.7.5.3. created `Date Invoice::created [private]`

Létrehozás dátuma

4.7.5.4. toBePaid `double Invoice::toBePaid [private]`

Fizetendő összeg.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- Invoice.h
- Invoice.cpp

4.8. String osztályreferencia

`#include <String.h>`

Publikus tagfüggvények

- size_t size () const
  Visszaadja a string hosszát.
- String ()
- const char * c_str () const
  C99 típusú string visszaadása.
- char * c_str ()
- String (size_t siz)
  Konstruktor méretből.
- String (char ch)
  Konstruktor karakterből.
- String (const char *p)
  Konstruktor egy C99 típusú stringből.
- String (const String &s1)
  Másoló konstruktor.
- virtual ~String ()
  Destruktor.
- bool operator== (const String &rhs) const
  Egyenlőség operátor.
- String & operator= (const String &rhs_s)
  Értékadó operátor.
- String operator+ (const String &rhs_s) const
  Két String objektumot összefűt.

Privát attribútumok

- char * pData
  Adat pointer.
- size_t len
  Lezáró nulla nélküli hossz.

Barátok

- std::ostream & operator<< (std::ostream &os, String p)
  Stream operator.

4.8.1. Konstruktorok és destruktorok dokumentációja

4.8.1.1. String() [1/5] String::String ( ) [inline]

4.8.1.2. String() [2/5] String::String (size_t siz)

Konstruktor méretből.

4.8.1.3. String() [3/5] String::String (char ch)

Konstruktor karakterből.

Konstruktor karakterből

Paraméterek

ch	- karakter, amiből létrehozzuk a stringet.

4.8.1.4. String() [4/5] String::String (const char * p)

Konstruktor egy C99 típusú stringből.

Konstruktor C99 típusú stringből.

Ez a default konstruktor is

Paraméterek

p	- C99 típusú string

4.8.1.5. String() [5/5] String::String (const String & a)

Másoló konstruktor.

Paraméterek

s1	- String, amit másolunk.

4.8.1.6. ~String() virtual String::~String ( ) [inline], [virtual]

Destruktor.

4.8.2. Tagfüggvények dokumentációja

4.8.2.1. c_str() [1/2] char * String::c_str ( ) [inline]

4.8.2.2. c_str() [2/2] const char * String::c_str ( ) const [inline]

C99 típusú string visszaadása.

Visszatérési érték

const char*

4.8.2.3. operator+() String String::operator+ (const String & *rhs_s*) const

Két String objektumot összefűt.

Paraméterek

rhs_s	- A jobb oldali string

Visszatérési érték

String - Az új összefűzött, módosított String .

4.8.2.4. operator=() String & String::operator= (const String & *a*)

Értékadó operátor.

Paraméterek

rhs_s	- A jobb oldali String objektum referenciája.

Visszatérési érték

String& A bal oldal módosított String referenciája.

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

Egyenlőség operátor.

Megvizsgálja, hogy két String megegyezik-e.

Paraméterek

rhs	- jobb oldali String objektum referenciája.

Visszatérési érték

true , ha megegyeznek.

false , ha nem egyeznek meg.

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

Visszaadja a string hosszát.

Visszatérési érték

size_t A string hossza

4.8.3. Barát és kapcsolódó függvények dokumentációja

4.8.3.1. operator<< `std::ostream & operator<< ( std::ostream & os, String p ) [friend]`

Stream operator.

Paraméterek

os	- output stream
p	- Kiírandó String

Visszatérési érték

std::ostream& - stream

4.8.4. Adattagok dokumentációja

4.8.4.1. len `size_t String::len [private]`

Lezáró nulla nélküli hossz.

4.8.4.2. pData `char* String::pData [private]`

Adat pointer.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- String.h
- String.cpp

4.9. Tariffs osztályreferencia

Ez a fájl tartalmazza a tarifákat tartalmazó Tariffs osztály deklarációit.

`#include <Tariffs.h>`

Statikus publikus attribútumok

- static double residental_16 =0

  Ez a fájl tartalmazza a tarifák inicializálását.
- static double residental_32 =0
- static double corporate_2ph_32 =0
- static double corporate_2ph_63 =0
- static double corporate_2ph_128 =0
- static double corporate_3ph_32 =0
- static double corporate_3ph_63 =0
- static double corporate_3ph_128 =0
- static double usage_fee =0

4.9.1. Részletes leírás

Ez a fájl tartalmazza a tarifákat tartalmazó Tariffs osztály deklarációit.

4.9.2. Adattagok dokumentációja

4.9.2.1. corporate_2ph_128 `double Tariffs::corporate_2ph_128 =0 [static]`

Vállalati | 2 fázis | 128A

4.9.2.2. corporate_2ph_32 double Tariffs::corporate_2ph_32 =0 [static]

Vállalati | 2 fázis | 32A

4.9.2.3. corporate_2ph_63 double Tariffs::corporate_2ph_63 =0 [static]

Vállalati | 2 fázis | 63A

4.9.2.4. corporate_3ph_128 double Tariffs::corporate_3ph_128 =0 [static]

Vállalati | 2 fázis | 128A

4.9.2.5. corporate_3ph_32 double Tariffs::corporate_3ph_32 =0 [static]

Vállalati | 3 fázis | 32A

4.9.2.6. corporate_3ph_63 double Tariffs::corporate_3ph_63 =0 [static]

Vállalati | 3 fázis | 63A

4.9.2.7. residental_16 double Tariffs::residental_16 =0 [static]

Ez a fájl tartalmazza a tarifák inicializálását.

Lakossági | 16A

Mindegyik 0-ra inicailizálódik, mert a fájlok betöltésekor állítódnak be az értékeik

4.9.2.8. residental_32 double Tariffs::residental_32 =0 [static]

Lakossági | 32A

4.9.2.9. usage_fee double Tariffs::usage_fee =0 [static]

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- Tariffs.h
- Tariffs.cpp

5. Fájlok dokumentációja

5.1. Address.cpp fájlreferencia

Ez a fájl tartalmazza az Address osztály definíciót.

```
#include "Address.h"
```

5.1.1. Részletes leírás

Ez a fájl tartalmazza az Address osztály definíciót.

Szerző

Gutási Ádám

Dátum

2022-05-15

5.2. Address.h fájlreferencia

Ez a fájl tartalmazza az Address osztály deklarációját és inline függvényeit.

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

Osztályok

- class Address

5.2.1. Részletes leírás

Ez a fájl tartalmazza az Address osztály deklarációját és inline függvényeit.

Szerző

Gutási Ádám

Dátum

2022-05-15

5.3. Address.h

Ugrás a fájl dokumentációjához.

```
1 #ifndef ADDRESS_HEADER
2 #define ADDRESS_HEADER
10 #include <iostream>
11 #include "String.h"
12
13 class Address{
14     private:
15         String town;
23         String street;
24
25         int house;
26         int apartment;
27     public:
28         Address() {};
29         Address(String t, String str,int h,int apt=1): town(t), street(str), house(h), apartment(apt){};
30
31         String& getStreet();
32         String& getTown();
33         int getHouse();
34         int getApartment();
36         bool operator==(Address& rhs);
37 };
38
39 #endif
```

5.4. Array.hpp fájlreferencia

A generikus dinamikus tömb megvaósítását tartalmazó fájl.

```
#include <stdexcept>
```

Osztályok

- class Array< T >
  Generikus dinamikus tömb.

Makródefiníciók

- #define DEBUG 1

Függvények

- template<typename MSG >
  void debug (std::ostream &os, MSG message)
  Debug információ kiírása egy streamre.

5.4.1. Részletes leírás

A generikus dinamikus tömb megvaósítását tartalmazó fájl.

Szerző

Gutási Ádám

Dátum

2022-05-15

5.4.2. Makródefiníciók dokumentációja

5.4.2.1. DEBUG `#define DEBUG 1`

5.4.3. Függvények dokumentációja

5.4.3.1. debug() `template<typename MSG >`

```
void debug (
            std::ostream & os,
            MSG message )
```

Debug információ kiírása egy streamre.

Azért itt lett definiíálva, mert az Array minden olyan osztályban includoeolva van, ahol kell debug információ.

Csak a DEBUG makró definiálása esetén működik

Sablon paraméterek

MSG	- üzenet típusa

Paraméterek

os	- stream, amire kiírjuk a debug információt.
message	- streamre kiírandó üzenet

5.5. Array.hpp

Ugrás a fájl dokumentációjához.

```
1
9 #ifndef ARRAY_H
10 #define ARRAY_H
11
12 #include <stdexcept>
13
26 template <typename MSG>
27 void debug(std::ostream& os, MSG message){
28     #ifdef DEBUG
29         os « message « std::flush;
30     #endif
31 }
32 #define DEBUG 1
33
39 template <class T>
40 class Array{
41     T* data;
42     size_t used;
43     public:
44         Array(size_t Siz=0): used(Siz){data=new T[used];}
45         virtual ~Array(){delete[] data;}
52         void add(T& newElement){
53             T* tmp=new T[this->used+1]; // ideiglenes, 1 mérettel nagyobb tároló
54             for(size_t i=0;i<this->used;i++){
55                 tmp[i]=this->data[i];
56             }
57         tmp[this->used]=newElement; // ideiglenes térolóhoz adjuk
58         delete[] this->data; // Array tárolójának törlése
59         this->data=tmp; // data pointer átállítása
60         used++; // használt méret nő.
61         }
62
73         T& get(size_t index) const{
74             if(index>=used){throw(std::out_of_range("Indexelesi hiba get"));}
75             return this->data[index];
76         }
77
83         size_t size() const{return used;}
84
90         T* begin() const{return data;}
91
100         T* end() const{
101             // pl.: 1 elem van -> data elején van az adat, 1-el előrebb kell mutasson a pointer!
102             // Ha 0 elem van használatban, akkor önmmagára fog mutatni, nem fut.
103             return data+(used);
104         }
105
112         int isElement(T& e) const{
113                 int i=0;
114                 for (T* dat=begin(); dat!=end(); dat++,i++) {
115                     debug(std::cout,e); debug(std::cout,"=="); debug(std::cout, *dat);
      debug(std::cout,"?\n");
116                     if ((*dat==e)) {
117                         debug(std::cout, "@"); debug(std::cout,i);
118                         return i;
119                     }
120                     debug(std::cout,"N");
121                 }
122                 return -1;
123         }
131         void del(size_t indx){
132             if(indx>=used){throw(std::out_of_range("Indexelesi hiba - del"));}
```

```
133             T* tmp= new T[used-1]; size_t tmpAt=0;
134             for(size_t pos=0;pos<used;pos++){
135                 if(pos!=indx){
136                     tmp[tmpAt++]=data[pos];
137                 }
138             }
139             used--;
140             this->data=tmp;
141 
142         }
143 
144 
151         void del(T& e){
152             int flag=isElement(e);
153 
154             debug(std::cout, "flag:");
155             debug(std::cout, flag);
156             debug(std::cout, "\n");
157 
158             if(flag==-1){
159                 throw(std::logic_error("Element not in array."));
160             }else{
161                del(flag);
162             }
163         }
164 
174         T& operator[](size_t index) const{
175             // a used a használatban levő Elemek számát adja meg.
176             if(index>=used){throw(std::out_of_range("Indexelesi hiba - operator[]"));}
177             return data[index];
178         }
179 
186         Array& operator=(const Array& rhs){
187             delete[] data;
188             data= new T[rhs.size()];
189             for(size_t i=0;i<used;i++){
190                 data[i]=rhs[i];
191             }
192             return *this;
193         }
194 
195 };
196 
197 #endif
```

5.6. Client.cpp fájlreferencia

Ez a fájl tartalmazza a Client osztály deklarációját és inline függvényeit.

```
#include "Client.h"
```

Függvények

- std::ostream & operator<< (std::ostream &os, Client &c)

  stream operátor

5.6.1. Részletes leírás

Ez a fájl tartalmazza a Client osztály deklarációját és inline függvényeit.

Szerző

Gutási Ádám

Dátum

2022-05-15

5.6.2. Függvények dokumentációja

5.6.2.1. operator<<() `std::ostream & operator<< ( std::ostream & os, Client & c )`

stream operátor

Paraméterek

os	- stream
c	- Ügyfél obejmtum referenciája

Visszatérési érték

std::ostream& - stream

5.7. Client.h fájlreferencia

Ez a fájl tartalmazza a Client osztály deklarációját és inline függvényeit.

```
#include <iostream>
#include "Address.h"
#include "Date.h"
#include "String.h"
#include "Client.h"
#include "Invoice.h"
#include "Array.hpp"
#include "Consumption_announcement.h"
```

Osztályok

- class Client

5.7.1. Részletes leírás

Ez a fájl tartalmazza a Client osztály deklarációját és inline függvényeit.

Szerző

Gutási Ádám

Dátum

2022-05-15

5.8. Client.h

Ugrás a fájl dokumentációjához.

```
1 #ifndef CLIENT_H
2 #define CLIENT_H
3
13 #include <iostream>
14 #include "Address.h"
15 #include "Date.h"
16 #include "String.h"
17 #include "Client.h"
18 #include "Invoice.h"
19 #include "Array.hpp"
20 #include "Consumption_announcement.h"
21
22
23 class Client{
24     friend class Controller;
25     private:
26         int id;
27         String firstName;
28         String lastName;
29         Date born;
30         Address resAddress;
31         String mobile;
32         String e_mail;
33         String taxNumber;
34         int electricMeter_last;
36         bool type;
38         int phases;
39         int strength;
42         double balance;
44     public:
45         Consumption_announcement announcement;
46         Array<Invoice> archivedInvoices;
47         Array<Invoice> pendingInvoices;
49         Client() {}; // default ctor
50         Client(int id, String lN, String fN, Date b, Address res, String m,
51         String em, String taxN, bool type, int phases, int strength): id(id), firstName(fN),
       lastName(lN), born(b), resAddress(res),
52         mobile(m), e_mail(em), taxNumber(taxN), electricMeter_last(0), type(type), phases(phases),
       strength(strength), balance(0){
53
54         }
55
56         Client& operator=(Client& rhs);
58         size_t ClientSize();
59
60         void addFunds(double moneyVal);
61         double getBalance() const;
62         int getId() const;
63         String getPhoneNumber() const;
64         String getEMail() const;
65         Date& getDate();
66         String getfirstName() const;
67         String getlastName() const;
68         Address& getAddress();
70         int getPhases() const;
71         int getStrength() const;
74         bool getType() const;
76         String getTN() const;
77         int getElectricMeterVal() const;
79         double getDebtval() const;
81         void pay_Pending_Invoices();
83         friend std::ostream& operator<<(std::ostream& os, Client& c);
84         protected:
85             void modify_electricMeter(int amt);
88 };
89
90 #endif
```

5.9. Consumption_announcement.cpp fájlreferencia

Consumption_announcement osztály tagfüggvényeinek definíiót tartalmazó fájl.

```
#include "Consumption_announcement.h"
```

5.9.1. Részletes leírás

Consumption_announcement osztály tagfüggvényeinek definíiót tartalmazó fájl.

Szerző

Gutási Ádám

Dátum

2022-05-15

5.10. Consumption_announcement.h fájlreferencia

Ez a fájl tartalmazza a Consumption_announcement osztály deklarációját és inline függvényeit.

```
#include "Date.h"
```

Osztályok

- class Consumption_announcement

5.10.1. Részletes leírás

Ez a fájl tartalmazza a Consumption_announcement osztály deklarációját és inline függvényeit.

Szerző

Gutási Ádám

Dátum

2022-05-15

5.11. Consumption_announcement.h

Ugrás a fájl dokumentációjához.

```
1 #ifndef CANNOUNCEMENT_H
2 #define CANNOUNCEMENT_H
3
11 #include "Date.h"
12
13 class Consumption_announcement{
14     private:
15         Date date;
16         int electricMeterVal;
17     public:
18         Consumption_announcement() :electricMeterVal(-1){};
19         Consumption_announcement(Date d, int emVal): date(d),
20         electricMeterVal(emVal){};
21
22         Date& getDate();
23         int get_EM_val();
29         void Reset();
30
31
32         bool operator==(Consumption_announcement& rhs);
34 };
35 #endif
```

5.12. Controller.cpp fájlreferencia

Az MVM rendszert kezelő osztály, amely a nyilvántartást is tartalmazza.

```
#include "Controller.h"
#include "math.h"
```

5.12.1. Részletes leírás

Az MVM rendszert kezelő osztály, amely a nyilvántartást is tartalmazza.

Szerző

Gutási Ádám

Ezzel a megoldással a program minimális módosítással egyszerre akár több, egymástól teljesen elválasztott adathalmazt is tud kezelni, szimplán egy új Controller osztály meghívásával.

Dátum

2022-05-15

5.13. Controller.h fájlreferencia

Ez a fájl tartalmazza a Controller osztály- és tagfüggvényeinek deklarációját.

```
#include <fstream>
#include "Client.h"
#include "Array.hpp"
#include "Tariffs.h"
#include "Consumption_announcement.h"
```

Osztályok

- class Controller

  Az MVM rendszert kezelő osztály, amely a nyilvántartást is tartalmazza.

5.13.1. Részletes leírás

Ez a fájl tartalmazza a Controller osztály- és tagfüggvényeinek deklarációját.

Szerző

Gutási Ádám

Dátum

2022-05-15

5.14. Controller.h

Ugrás a fájl dokumentációjához.

```
1 #ifndef CONTROLLER_H
2 #define CONTROLLER_H
3
12 #include <fstream>
13 #include "Client.h"
14 #include "Array.hpp"
15 #include "Tariffs.h"
16 #include "Consumption_announcement.h"
17
18
23 class Controller {
24
25     private:
26         Array<Client> clients;
27         // Tariffs tariffs; A tarfa absztrakt osztály. Nem kell létrehozni.
28     public:
29         Controller(){};
30         void loadData(char const* CData, char const* Invoices,
31         char const* Invoices_pending, char const* Tariffs, char const* CAnnFile);
32         void saveData(char const* CData, char const* Invoices,
33         char const* Invoices_pending_file, char const* CAnnFile);
34
35         void create_Invoices(Date& todayDate);
36         //void create_Invoice();
37         void newClient(Client& c);
38         void announceConsumption(Client&c,int emVal, Date& d);
39         Client& getClient(size_t id);
40
41         double calculate_toBePaid(Client& c);
42
43         size_t clientsCount();
44
45         /* Azért nincs a dtorban a mentés meghívása,
46         ,mert például a számlázás, és fizetés után az adatok megváltoznak,
47         a következő futáskor a teszt hibásan futna le.*/
48 };
49
50 #endif
```

5.15. Date.h fájlreferencia

A dátum osztály deklarációit tartalmazó fájl.

`#include <iostream>`

Osztályok

- class Date

Függvények

- std::ostream & operator<< (std::ostream &os, Date &d)

  Kiír az ostream-re.

5.15.1. Részletes leírás

A dátum osztály deklarációit tartalmazó fájl.

Szerző

Gutási Ádám

Dátum

2022-05-15

5.15.2. Függvények dokumentációja

5.15.2.1. operator<<() std::ostream & operator<< (std::ostream & os, Date & d) [inline]

Kiír az ostream-re.

Paraméterek

os	- ostream objektum
d	- kiírandó dátum.

Visszatérési érték

std::ostream&

5.16. Date.h

Ugrás a fájl dokumentációjához.

```
1 #ifndef DATE_H
2 #define DATE_H
10 #include <iostream>
11
12 class Date{
13     private:
14         unsigned int year;
15         unsigned int month;
16         unsigned int day;
17     public:
18         friend std::ostream& operator«(std::ostream& os, Date& d);
19         Date() {};
20         Date(unsigned int y,unsigned int m, unsigned int d): year(y), month(m), day(d){};
21
27         unsigned int getYear(){return year;};
28
34         unsigned int getMonth(){return month;};
35
41         unsigned int getDay(){return day;};
42
50         inline bool operator==(Date& rhs){
51             if(year==rhs.getYear() && month==rhs.getMonth() && day==rhs.getDay()){
52                 return true;
53             }
54             return false;
55         }
56
64         inline bool operator<(Date& rhs){
65             if((year<rhs.getYear()) || ((year==rhs.getYear() && (month<rhs.getMonth())))
66             || ((year==rhs.getYear() && (month==rhs.getMonth()) && (day<rhs.getDay())))){
67                 return true;
68             }else{
69                 return false;
70             }
71         }
72         inline bool operator<=(Date& rhs){return (operator<(rhs) || operator==(rhs));}
73
81         inline bool operator>(Date& rhs){
82             if((year>rhs.getYear()) || ((year==rhs.getYear() && (month>rhs.getMonth())))
83             || ((year==rhs.getYear() && (month==rhs.getMonth()) && (day>rhs.getDay())))){
84                 return true;
85             }else{
86                 return false;
87             }
88         }
89
97         inline bool operator>=(Date& rhs){return (operator>(rhs) || operator==(rhs));}
```

```
98
99
100 };
101
109 inline std::ostream& operator«(std::ostream& os, Date& d){
110     os « d.getYear() « "." « d.getMonth() « "." « d.getDay();
111     return os;
112 }
113
114 #endif
```

5.17. Invoice.cpp fájlreferencia

Az Invoice osztályhoz tartozó tagfüggvények definíciói.

```
#include "Invoice.h"
```

Függvények

- std::ostream & operator<< (std::ostream &os, Invoice &i)

  stream operátor

5.17.1. Részletes leírás

Az Invoice osztályhoz tartozó tagfüggvények definíciói.

Szerző

Gutási Ádám

Dátum

2022-05-15

5.17.2. Függvények dokumentációja

5.17.2.1. operator<<() `std::ostream & operator<< ( std::ostream & os, Invoice & i )`

stream operátor

Paraméterek

os	- stream
i	- Számla

Visszatérési érték

std::ostream& - stream

5.18. Invoice.h fájlreferencia

Ez a fájl tartalmazza az Invoice osztály deklarációját és inline függvényeit.

```
#include "Date.h"
#include "Consumption_announcement.h"
```

Osztályok

- class Invoice

  Egy számla osztálya.

5.18.1. Részletes leírás

Ez a fájl tartalmazza az Invoice osztály deklarációját és inline függvényeit.

Szerző

Gutási Ádám

Dátum

2022-05-15

5.19. Invoice.h

Ugrás a fájl dokumentációjához.

```
1 #ifndef INVOICE_H
2 #define INVOICE_H
3
11 #include "Date.h"
12 #include "Consumption_announcement.h"
13
18 class Invoice{
19     private:
20         Date created;
21         Consumption_announcement announcement;
22         int consumptionAmt;
23         double toBePaid;
25     public:
26         Invoice() : consumptionAmt(0), toBePaid(0){};
27
35         Invoice(Date c, Consumption_announcement& cAnnounce, int cAmt): created(c),
36         announcement(cAnnounce), consumptionAmt(cAmt), toBePaid(0) {};
37
38         Date& getCreated();
39         Consumption_announcement& getCAnn();
40         void set_toBePaid(double what);
41         int getConsumptionAmt() const;
42         double get_toBePaid() const;
43         friend std::ostream& operator<<(std::ostream& os, Invoice& c);
45         bool operator==(Invoice& rhs);
46 };
47
48 #endif
```

5.20. main.cpp fájlreferencia

Ez a fájl a MVM projekt unit testje. A tesztelés a gtest_lite könyvtárral történik.

```
#include <iostream>
#include <stdexcept>
#include <sstream>
#include "Controller.h"
#include <fstream>
#include "gtest_lite.h"
#include "memtrace.h"
```

Függvények

- int main ()

5.20.1. Részletes leírás

Ez a fájl a MVM projekt unit testje. A tesztelés a gtest_lite könyvtárral történik.

Szerző

Gutási Ádám

Szúrópróbaszerű adatellenőrzések történnek, tömbhosszak összevetése az elvárt eredményekkel, stb.

Dátum

2022-05-15

5.20.2. Függvények dokumentációja

5.20.2.1. main() `int main ( void )`

5.21. mvm_with_menu.cpp fájlreferencia

Ez a fájl a parancssori, végfelhasználói verzió. A felhasználó egy menün keresztül tudja vezérelni a programot, adatokat pedig a terminálon keresztül tud bevinni.

```
#include <iostream>
#include <fstream>
#include "Controller.h"
#include "Array.hpp"
#include "memtrace.h"
```

Függvények

- void add_newClient (Controller &Ctrl, int incr)
- int main (void)

5.21.1. Részletes leírás

Ez a fájl a parancssori, végfelhasználói verzió. A felhasználó egy menün keresztül tudja vezérelni a programot, adatokat pedig a terminálon keresztül tud bevinni.

Szerző

Gutási Ádám

A Debug információ a DEBUG makró befiniálásával kapcsolható be.

Dátum

2022-05-15

5.21.2. Függvények dokumentációja

5.21.2.1. add_newClient()

```
void add_newClient (
        Controller & Ctrl,
        int incr )
```

5.21.2.2. main()

```
int main (
        void  )
```

5.22. README.md fájlreferencia

5.23. String.cpp fájlreferencia

Tartalmazza a String osztály definícióit.

```
#include <iostream>
#include <cstring>
#include "String.h"
```

Függvények

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

  Stream operator.

5.23.1. Részletes leírás

Tartalmazza a String osztály definícióit.

Szerző

Gutási Ádám

Dátum

2022-05-15

5.23.2. Függvények dokumentációja

5.23.2.1. operator<<() `std::ostream & operator<< ( std::ostream & os, const String s )`

Stream operator.

Paraméterek

os	- output stream
p	- Kiírandó String

Visszatérési érték

std::ostream& - stream

5.24. String.h fájlreferencia

Ez a fájl taartalmazza a String osztály deklarációit, és az inline függvényeket.

```
#include <iostream>
```

Osztályok

- class String

Függvények

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

  Karakterhez sztringet fűz.

5.24.1. Részletes leírás

Ez a fájl taartalmazza a String osztály deklarációit, és az inline függvényeket.

Szerző

Gutási Ádám

Dátum

2022-05-15

5.24.2. Függvények dokumentációja

5.24.2.1. operator+()

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

Karakterhez sztringet fűz.

Paraméterek

ch	- lvalue
str	- rvalue

Visszatérési érték

String - Új, összefűzött Stringgel tér vissza.

5.25. String.h

Ugrás a fájl dokumentációjához.

```
1 #ifndef STRING_H
2 #define STRING_H
12 #include <iostream>
13
14 class String {
15     char *pData;
16     size_t len;
17 public:
23     size_t size() const { return len; }
24
25     String() :pData(0), len(0) {}
26
32     const char* c_str() const { return pData;}
33     char* c_str() { return pData;}
34
36     String(size_t siz);
37
40     String(char ch);
41
48     String(const char *p); // Ugyanaz, mint a "".
49
55     String(const String& s1);
56
58     virtual ~String() { delete[] pData; }
```

```
59
68      bool operator==(const String& rhs)const;
69
70
77      String& operator=(const String& rhs_s);
78
85      String operator+(const String& rhs_s) const ;
86
94      friend std::ostream& operator«(std::ostream& os, String p);
95 };
103 inline String operator+(char ch, const String& str) { return String(ch) + str; }
104
105 #endif
```

5.26. Tariffs.cpp fájlreferencia

```
#include "Tariffs.h"
```

5.27. Tariffs.h fájlreferencia

Osztályok

- class Tariffs

  Ez a fájl tartalmazza a tarifákat tartalmazó Tariffs osztály deklarációit.

5.28. Tariffs.h

Ugrás a fájl dokumentációjához.

```
1 #ifndef TARIIFS_H
2 #define TARIIFS_H
9 class Tariffs{
10      public:
11          static double residental_16;
12          static double residental_32;
13          static double corporate_2ph_32;
14          static double corporate_2ph_63;
15          static double corporate_2ph_128;
16          static double corporate_3ph_32;
17          static double corporate_3ph_63;
18          static double corporate_3ph_128;
19          static double usage_fee;
20 };
21 #endif
```

Tárgymutató