HÁZI FELADAT

Programozás alapjai 2.

Végleges

Drahos Zsolt
UCZFU3

2020. május 13.

TARTALOM

1. Feladat

Telefonkönyv

Tervezze meg egy telefonkönyv alkalmazás egyszerűsített objektummodelljét, majd valósítsa azt meg! A telefonkönyvben kezdetben az alábbi adatokat akarjuk tárolni, de később bővíteni akarunk:

- Név (vezetéknév, keresztnév)
- becenév
- munkahelyi szám
- privát szám
- lakcím (ország, irányítószám, város, utca, házszám)

A telefonkönyv képes új névjegyek felvételére, törlésére, listázásra, keresésre.

2. Feladatspecifikáció

A feladat egy telefonkönyv alkalmazás egyszerűsített objektummodelljének az elkészítése. Mivel a telefonkönyvben nem definiált mennyiségű adatot akarunk tárolni, ezért az adatok dinamikusan lesznek elhelyezve.

A program képes új adatok felvételére szabványos bemenetről is. Továbbá képes a felvett adatok törlésére és listázására. Ezen felül a felhasználó tud az adatok között keresni, akár név, becenév, lakcím vagy telefonszám alapján is. Amennyiben bármi hiba történik a funkciók működése közben akkor hibát fog dobni.

Minden adat egyaránt tartalmazhat szöveget és számot, melyek dinamikusan lesznek tárolva, ezzel lehetősége nyílik a felhasználónak végtelen hosszú adatok felvételére.

A tesztelésére egy olyan programot készítek, melyek letesztelik a funkciókat felvett adatokkal és szabványos bemenetről beolvasott adatokkal is. Továbbá készítek egy olyan tesztet, ami direkt hibás adatot fog tartalmazni ezzel szemléltetve a kivételkezelést.

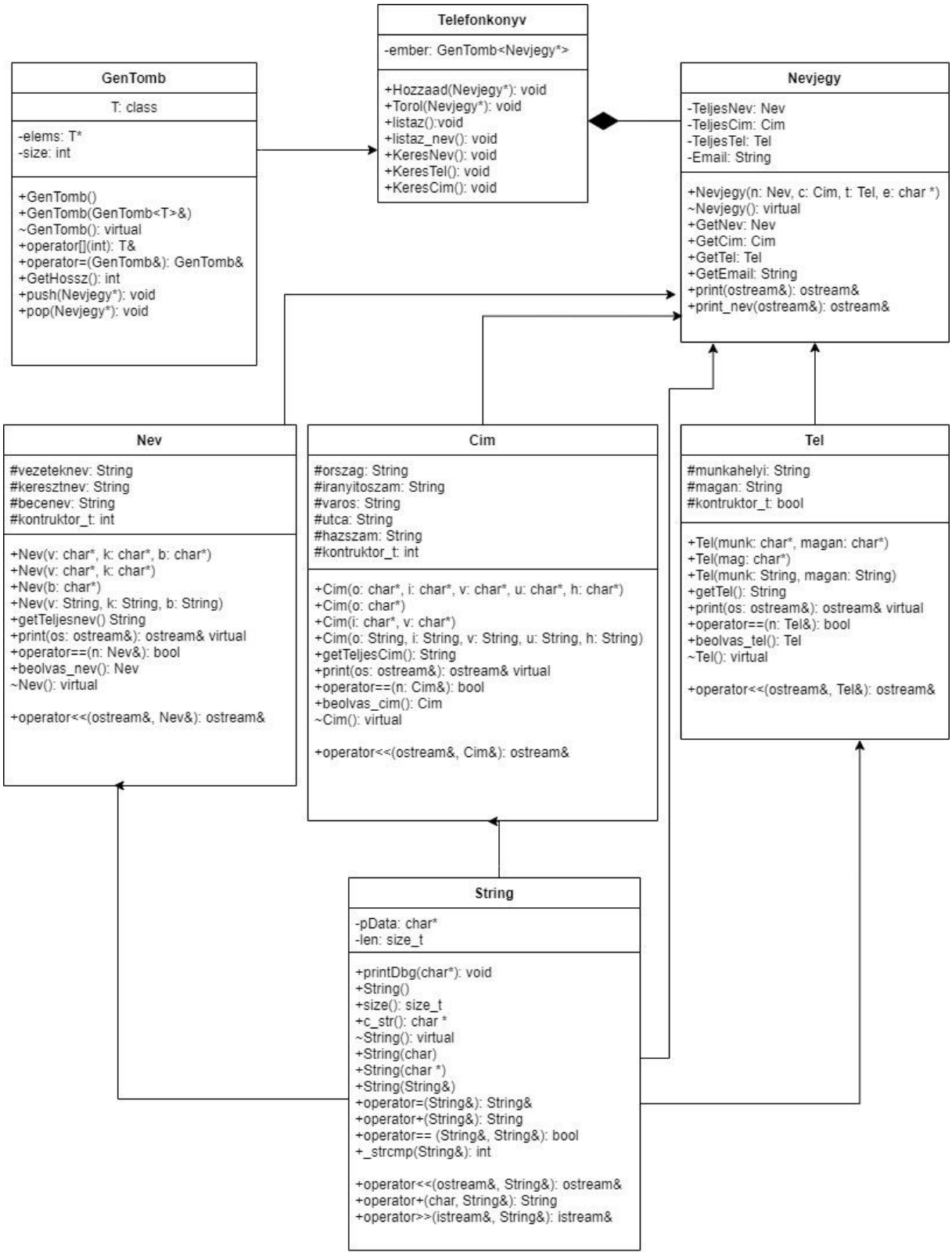
3. Terv

A feladat elkészítéséhez szükség van hét darab osztály objektumra, továbbá elengedhetetlen a tesztprogram elkészítése.

A telefonkönyv programot osztályokkal és sablonokkal fogom megvalósítani. A telefonkönyv adatait egy saját készítésű dinamikus tömb osztályban (`class GenTomb`) fogom tárolni, ami sablonként veszi át a telefonkönyv elemeinek típusát (`Nevjegy*`).
A névjegy osztályban lesznek elérhetőek a személy adatok, melyek a következők: Név (Vezetéknév, Keresztnév, Becenév), Cím (Ország, Irányítószám, Város, Utca, Házszám), Telefonszám (Magán, Munkahelyi) és E-mail cím. Ezek az adatok (kivéve az Email) külön osztályként fognak kapcsolódni a `Névjegy` osztályhoz.

A `Név`, `Cím`, `Tel` osztályban lévő adatok `String` típusúak mely egy saját készítésű osztály, ami dinamikusan tárolja a szöveget.

3.1. Objektum terv



4. Megvalósítás

A feladathoz elkészült hét osztály és négy tesztprogram, amikkel meg tudjuk hívni a program minden függvényét. A program elsősorban <iostream>-et használ. Az osztályok végleges függvényei az objektum terv fülnél látható. Tervezési lépéshez képest csak egy két függvény lett eltávolítva mert feleslegesen lettek deklarálva. Az osztályok a saját nevük által álnevezett .h és .cpp fájlokban találhatóak, a tesztprogram pedig a main.cpp fájlban lett elkészítve.

4.1. Osztályok bemutatása

Mint ahogy már az előbb írtam, hét osztály kellett a program elkészítéshez. Minden osztálynak a deklarációja és inline függvényei az <osztalynev>.h fájlban található, míg a többi függvény az <osztalynev>.cpp fájlban található. A tagfüggvények leírás itt csak felületes, részletesebb leírást a Mellékelt fülnél - a kódban található.

4.1.1. GenTomb osztály

A generikus tömb osztályban az adatok dinamikusan vannak tárolva, így a felhasználó tetszőleges számú névjegyet képes benne eltárolni. Nagyon hasonló a std::vector tárolóhoz.

Attributumok

- T * **elems** - *T osztály pointer.*
- int **size** - *elemek száma a tömbben*

Konstruktor / Destruktor

- **template<class T > GenTomb< T >::GenTomb ()[inline]**

Alapértelmezett Konstruktor size - automatikusan 0 értéket állít be lefoglalja a helyet dinamikusan

- **template<class T> GenTomb<T>::GenTomb (const GenTomb< T > & *s*)[inline]**

Másoló konstruktor helyet dinamikusan foglalja

s	- konstans T-ket tartalmazó referenciát vesz át

- **template<class T > GenTomb< T >::~GenTomb ()[inline]**

Destruktor.

Publikus tagfüggvények

- **GenTomb** ()
- **GenTomb** (const **GenTomb**< T > &s)
- **~GenTomb** () - *Destruktor.*
- T & **operator[]** (int index)- Indexelõ operátor
- **GenTomb** & **operator=** (T temp) - Értékadó operátor
- int **GetHossz** () - Függvény, ami visszatér a tömb hosszával
- void **push** (**Nevjegy** *n) - Függvény, ami hozzáad egy elemet a tömb végéhez
- void **pop** (**Nevjegy** *n) - Függvény, ami töröl egy elemet a tömbből. Vizsgálja, hogy nem lehet üres a tömb

4.1.2. Telefonkonyv osztály

A telefonkönyv osztályban egyetlen egy attribútum található, ami az ember névre hallgat. Ez a generikus tömb képes tárolni a Nevjegy osztály adatait. Itt találhatók a program főbb funkcionális függvényei, mint például a törlés listázás vagy a keresések.

Attribútumok

- **GenTomb<Nevjegy*> ember - *névjegy adatai, ami egy generikus tömbben van eltárolva***

Publikus tagfüggvények

- **void Hozzaad (Nevjegy *n) –** *Függvény, ami hozzáadja a tömbhöz az adott névjegyet.*
- **void Tororl (Nevjegy *n) –** *Függvény, ami kitörli az adott névjegyet a tömbbõl.*
- **void listaz (std::ostream &os) –** *Függvény, ami kilistázza az adatokat.*
- **void listaz_nev (std::ostream &os) -** *Függvény ami kilistázza csak a neveket.*
- **int darab () –** *Visszatér a névjegyek számával.*
- **void KeresNev (const Nev &n, std::ostream &os) –** *Függvény, ami megkeresi az adott nevű névjegyet.*
- **void KeresTel (const Tel &t, std::ostream &os) –** *Függvény, ami megkeresi az adott telefonszámú névjegyet.*
- **void KeresCim (const Cim &c, std::ostream &os) –** *Függvény, ami megkeresi az adott című névjegyet.*

4.1.3. Nevjegy osztály

A névjegy osztály tulajdonképpen egyesíti és összefogja a név, cím és telefonszám osztályokat. Ezek az osztályok segítségével jön létre egy névjegy típus, ami kerül bele a tömbbe.

Attribútumok

- Nev **TeljesNev** - Teljes név (Név konstruktortól függően teljes, vagy csak a becenév)
- Cim **TeljesCim -** Teljes cím (Cím konstruktortól függően az összes, vagy csak az ország, irányítószám, város)
- Tel **TeljesTel** - Teljes telefonszám (tel konstruktortól függően mindkettő, vagy csak a magán)
- String **Email**

Konstruktor / Destruktor

- **Nevjegy::Nevjegy (const Nev & *n*, const Cim & *c*, const Tel & *t*, String *e*)[inline]**

Konstruktor beállítja az attribútumokat

n	- nev
c	- cim
t	- telefonszam
e	- Email **String**

- **virtual Nevjegy::~Nevjegy ()[inline], [virtual]**

Destruktor.

Publikus taggfügvények

- **Nevjegy** (const **Nev** &n, const **Cim** &c, const **Tel** &t, **String** e)
- **Nev getNev** () - *Függvény, ami visszatér a Teljes Név értékkel*
- **Cim getCim** () - *Függvény, ami visszatér a Teljes Cim értékkel*
- **Tel getTel** () - *Függvény, ami visszatér a Teljes Telefonszám értékkel*
- virtual std::ostream & **print** (std::ostream &os) const - *Virtuális függvény ami kiírja az adatokat "\n"-el elválasztva ostreamre*
- virtual std::ostream & **print_nev** (std::ostream &os) const - *Virtuális függvény ami kiírja az csak a neveket*
- virtual **~Nevjegy** () - *Destruktor.*

4.1.4. Nev osztály

Attribútumok

- String **vezeteknev** - *Vezetéknév*
- String **keresztnev** - *Keresztnév*
- String **becenev** - *Becenév*
- int **kontruktor_t** - *Segédváltozó a konstruktor attribútumok számának meghatározására.*

Publikus tagfüggvények

- **Nev (const char *v, const char *k, const char *b) -** *Konstruktor*
- **Nev (const char *v, const char *k) -** *Konstruktor*
- **Nev (const char *b) -** *Konstruktor*
- **Nev (String v, String k, String b)**
- String **getTeljesnev () const -** *Név lekérdezése értékadás alapján Mind a három adattagnak van értéke akkor visszatér a teljes névvel (közepén becenévvel) Ha csak a vezetéknév és a keresztnév van megadva akkor csak azzal a kettõvel tér vissza És ha csak a becenevet adjuk meg akkor csak a becenévvel tér vissza*
- virtual **std::ostream & print (std::ostream &os) const -** *Attribútumok kiírása egy stream-re*
- bool **operator== (const Nev &n) const -** *Két név egyezõségét vizsgálja konstrukor konstruktor paramétereinek száma alapján*
- Nev **beolvas_nev () -** *Adatok beolvasása istreamrõl*
- virtual **~Nev ()-** *Virtuális destruktor.*

Konstruktor / Destruktor

- **Nev::Nev (const char * *v*, const char * *k*, const char * *b*)[inline]**

Konstruktor beállítja az attribútumokat

k	- keresztnév megnevezése
v	- vezetéknév megnevezése
b	- becenév megnevezése

-
-
- **Nev::Nev (const char * *v*, const char * *k*)[inline]**

Konstruktor beállítja az attribútumokat

v	- vezetéknév megnevezése
k	- keresztnév megnevezése

- **Nev::Nev (const char * *b*)[inline]**

Konstruktor beállítja az attribútumokat

b	- becenév megnevezése

- **Nev::Nev (String *v*, String *k*, String *b*)[inline]**

Konstruktor beállítja az attribútumokat

k	- keresztnév megnevezése - **String** típusú
v	- vezetéknév megnevezése - **String** típusú
b	- becenév megnevezése - **String** típusú

- **virtual Nev::~Nev ()[inline], [virtual]**

Virtuális destruktor.

4.1.5. Cím osztály

Attribútumok

- **String orszag -** *Ország*
- **String iranyitoszam -** *Irányírószám*
- **String varos -** *Város.*
- **String utca -** *Utca.*
- **String hazszam -** *Házszám*
- int **kontruktor_t -** *Segédváltozó a konstruktor attribútumok számának meghatározására.*

Konstruktor / Destruktor

- **Cim::Cim (const char * *o*, const char * *i*, const char * *v*, const char * *u*, const char * *h*)[inline]**

Konstruktor beállítja az attribútumokat

o	- ország megnevezése
i	- irányítószám megnevezése
v	- város megnevezése
u	- utca megnevezése
h	- házszám megnevezése

- **Cim::Cim (const char * *o*)[inline]**

Konstruktor beállítja az attribútumokat

o	- ország megnevezése

- **Cim::Cim (const char * *i*, const char * *v*)[inline]**

Konstruktor beállítja az attribútumokat

i	- irányítószám megnevezése
v	- város megnevezése

- **Cim::Cim (String *o*, String *i*, String *v*, String *u*, String *h*)[inline]**

Konstruktor beállítja az attribútumokat

o	- **String** típusú
i	- **String** típusú
v	- **String** típusú
u	- **String** típusú
h	- **String** típusú

- **virtual Cim::~Cim ()[inline], [virtual]**

Virtuális destruktor.

Publikus tagfüggvények

- **Cim** (const char *o, const char *i, const char *v, const char *u, const char *h)
- **Cim** (const char *o)
- **Cim** (const char *i, const char *v)
- **Cim** (**String** o, **String** i, **String** v, **String** u, **String** h)
- **String getTeljesCim** () const - *Teljes cím lekérdezése konstruktor paramétereinek száma alapján Minden adattag: teljes cím vesszõvel elválasztva Csak országgal Csak Irányírószámmal és városnévvel*
- virtual std::ostream & **print** (std::ostream &os) const - *Attribútumok kiírása egy stream-re*
- bool **operator==** (const **Cim** &c) const - *Két cím egyezõségét vizsgálja*
- **Cim beolvas_cim** () - *Adatok beolvasása istreamrõl*
- virtual **~Cim** ()- *Virtuális destruktor.*

4.1.6. Tel osztály

Attribútumok

- **String munkahelyi -** *Munkahelyi telefonszám*
- **String magan -** *Magán telefonszám*
- bool **kontruktor_t -** *Segédváltozó a konstruktor attribútumok számának meghatározására.*

Konstruktor / Destruktor

- Tel::Tel (const char * *munk*, const char * *mag*)[inline]

Konstruktor beállítja az attribútumokat

munk	- munkahelyi telefonszám megnevezése
mag	- magán telefonszám megnevezése

- Tel::Tel (const char * *mag*)[inline]

Konstruktor beállítja az attribútumokat

mag	- magán telefonszám megnevezése

- Tel::Tel (String *munk*, String *mag*)[inline]

Konstruktor beállítja az attribútumokat

munk	- **String** típusú
mag	- **String** típusú

- virtual Tel::~Tel ()[inline], [virtual]

Virtuális destruktor.

Publikus taggfügvények

- **Tel** (const char *munk, const char *mag)
- **Tel** (const char *mag)
- **Tel** (**String** munk, **String** mag)
- **String getTel** () const - *Telefonszám lekérdezése értékadás alapján Munkahelyi és magán telefonszámmal való visszatérés Illetve csak magán telefonszámmal való visszatérés*
- virtual std::ostream & **print** (std::ostream &os) const - *Attribútumok kiírása egy stream-re*
- bool **operator==** (const **Tel** &t) const - *Két Telefonszám egyezõségét vizsgálja konstruktor paramétereinek száma alapján*
- **Tel beolvas_tel** () - *Adatok beolvasása istreamrõl*
- virtual **~Tel** () - *Virtuális destruktor.*

4.1.7. String osztály

Az 5. laboron készített `string5.h` és `string5.cpp` fájl kiegészítése.

Attribútumok

- String ***pData** – *pointer*
- size_t **len** – *szó hossza lezáró nulla nélkül*

Publikus taggfügvények

- void **printDbg** (const char *txt="") const - *Kiírunk egy tetszõleges szöveget*
- **String** () - *Konstruktor:*
- size_t **size** () const - ***String** hosszával történõ visszatérés*
- const char * **c_str** () const - *Stringet visszaadó függvény*
- virtual **~String** () - *Destruktor.*
- **String** (char ch) - *Konstruktorok: egy char karakterbÅ‘l.*
- **String** (const char *p) - *egy nullával lezárt char sorozatból*
- **String** (const **String** &s1) – *Másoló konstruktor*
- **String** & **operator=** (const **String** &rhs) – *Értékadó operátor*
- **String operator+** (const **String** &rhs) const - *Összeadó operaátor - Két Stringet fûz össze*
- char & **operator[]** (unsigned int idx) - *[] operátor: egy megadott indexű elem REFERENCIÁJÁVAL térnek vissza*
- int **_strcmp** (const **String** &rhs) const - *Segéd strcmp függvény.*
- Friends bool **operator==** (const **String** &lhs, const **String** &rhs)== Összehasonlító operátor - *Két stringet hasonlít össze*

Konstruktor / Destruktor

- **String::String ()[inline]**

Konstruktor

- **String::~String ()[virtual]**

Destruktor.

- **String::String (char *ch*)**

Konstruktor: egy char karakterbõl

ch	- karakter

- **String::String (const char * *p*)**

egy nullával lezárt char sorozatból

Konstruktor: egy nullával lezárt char sorozatból

p	- C típusú string

- **String::String (const String & *s1*)**

Másló konstruktor.

s1	- **String**

5. Tesztelés

A teszteseteket az alapján készülte el, hogy be tudja mutatni a program összes funkcióját és meghívja az elkészített függvényeket. Ezért kellett négy darab tesztet létrehozni, hogy teljesen lefedje a program minden részét.

5.1. test_1

Az első teszt (`void test_1()`) adatok felvételét és törlését mutatja be. Az adatok felvétele a kódban történik ahogy a képen is látszik.

```
Telefonkonyv t1;
Nevjegy n1(Nev("Vezeteknev","Keresztnev","Becenev"),
           Cim("Orszag","Irany","Varos","Utca/ut","Hazszam"),
           Tel("1234567890","9876543210"), "email@email.com");
t1.Hozzaad(&n1);
```

Ezek után kilistázzuk az összes adatot és végül kitöröljük az adatokat.

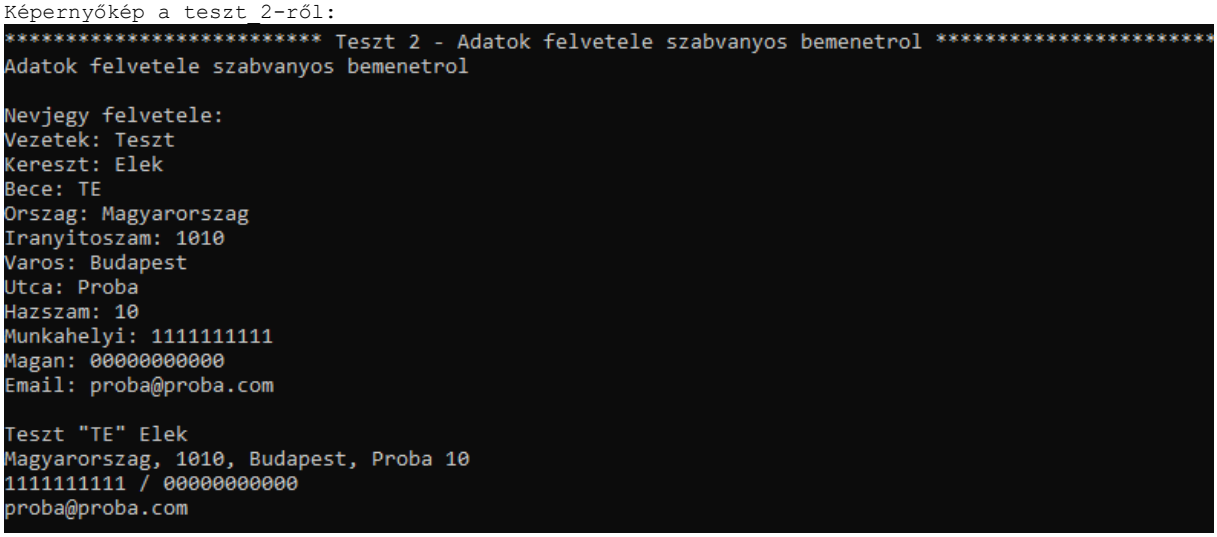
Képernyőkép a teszt_1-ről:

```
************************** Teszt 1 - Adatok felvetele kodban **************************

Adatok felvetele kodban

SIM kartyan tarolt nevjegyek: 4

1. nevjegy:
Vezeteknev "Becenev" Keresztnev
Orszag, Irany, Varos, Utca/ut Hazszam
1234567890 / 9876543210
email@email.com

2. nevjegy:
Teszt "TE" Elek
Magyarorszag, 1111, Budapest, Alma utca 10
06301234456 / 06306555002
elek@gmail.com

3. nevjegy:
Ferenci "Viva" Viva
Magyarorszag, 2234, Maglod, Sip utca 74
06295760940 / 06448599209
ferenci@viva.com

4. nevjegy:
asd "asd" asd
asd, asd, asd, asd asd asd
asd / asd
asd@asd.com

Teszt "TE" Elek nevu szemely torlese
1. nevjegy:
Vezeteknev "Becenev" Keresztnev

2. nevjegy:
Ferenci "Viva" Viva

3. nevjegy:
asd "asd" asd
```

5.2. test_2

A második teszt az adatok felvételét mutatja be szabványos bemenetről. Itt a felhasználó adhat meg egy tetszőleges névjegyet melyet hozzáadunk a telefonkönyvünkhöz. Miután felvettük az adatot, kilistázzuk az névjegyet, ezzel ellenőrizve, hogy sikerült hozzáadni a tömbhöz.

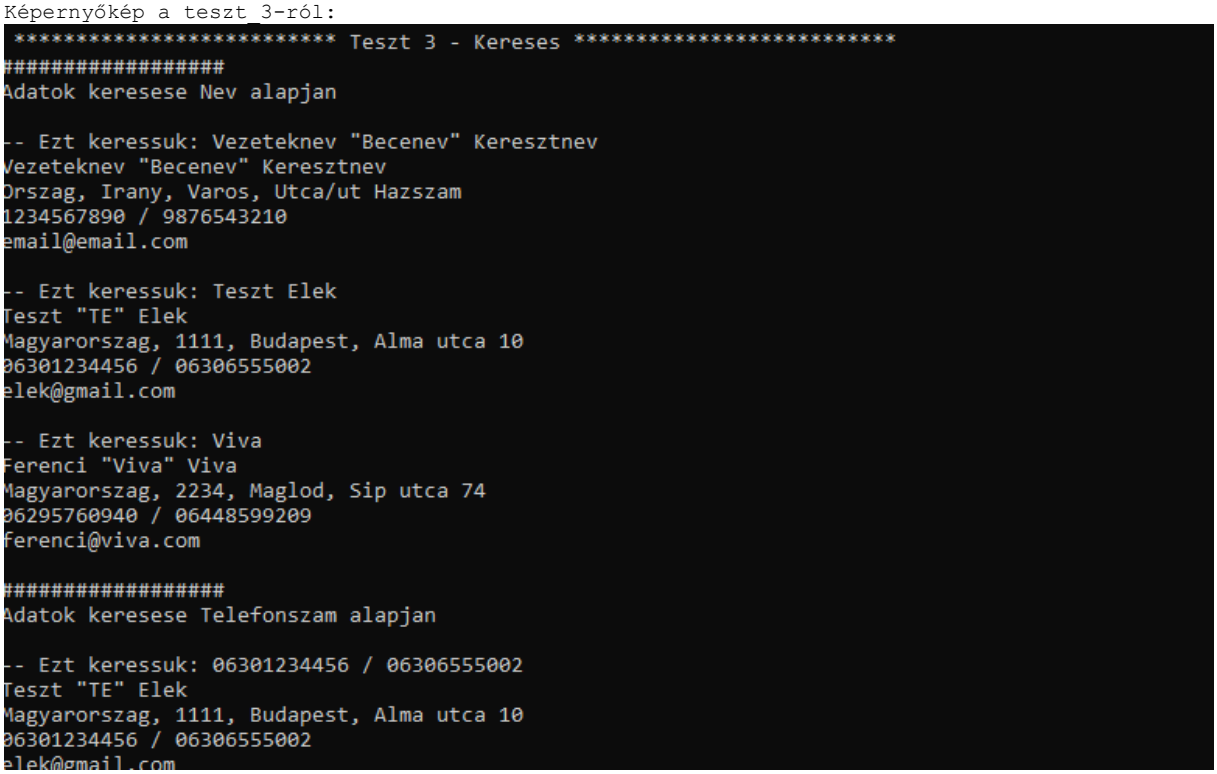
Képernyőkép a teszt_2-ről:



5.3. test_3

A harmadik tesztben a kódban felvesszük ugyan azokat a névjegyeket, amit az elsőben és meghívjuk a keres függvényeket. Ezek a függvények képesek nem csak teljes név/cím/telefonszám alapján keresni, hanem becenév/ország vagy akár magán telefonszám alapján is.

Képernyőkép a teszt_3-ról:



5.4. test_4

A negyedik teszt pedig a lehetséges hibák és azok kezelését mutatja be. Elsődleges probléma akkor léphet fel, ha üres tömbből szeretnék törölni adatot. Ekkor a program automatikusan szól, hogy „`Ures a tomb`”. Továbbá a program figyel arra is, hogy ha olyan adatot szeretnénk törölni, ami nincs benne a telefonkönyvben.

Képernyőkép a teszt 4-ről:

```
************************** Teszt 4 - Hibas adatok **************************
Adat torlese egy ures tombbol:
ERROR
Olyan adat keresese ami nincs benne a tombben:
-- Ezt keressuk: Nincs, Nincs
Nincs ilyen cim az adatbazisban!

Process returned 0 (0x0)   execution time : 232.211 s
Press any key to continue.
```

5.5. Memóriakezelés tesztje

A memóriakezelést a memtrace modullal végeztem, melyet a https://git.ik.bme.hu/Prog2/ell_feladat/Test oldalról töltöttem le. A modul működésének érdekében a `memtrace.h` fájlt minden fájlban include-oltam. A modul nem jelzett memóriaszivárgást.

5.6. Lefedettségi teszt

A tesztek a program minden fő ágát lefedték, kivéve olyan else / if ágakba nem lépett be, ahol úgy gondoltam, hogy elég csak egyszer meghívni a főprogramban. Például amikor nincs benne a tömbben vagy amikor nem találja meg az adott nevet / címet / telefonszámot.

```
108.        }
109.    }
110.    if (db == 0)
111.    {
112.        cout << "Nincs ilyen telefonszam az adatbazisban!" << endl;
113.    }
114. }
```

6. Mellékletek

6.1. cim.cpp

```
#include "cim.h"
#include "memtrace.h"
/// << operátor ami kiírja a Teljes Címet ostreamre
std::ostream& operator<<(std::ostream& os, const Cim& s0)
{
    os << s0.getTeljesCim();
    return os;
}
```

6.2. cim.h

```
#ifndef CIM_H
#define CIM_H
#include "string.h"
#include "memtrace.h"

class Cim
{
protected:
    String orszag;                  ///< Ország
    String iranyitoszam;            ///< Irányírószám
    String varos;                   ///< Város
    String utca;                    ///< Utca
    String hazszam;                 ///< Házszám
    int kontruktor_t;               ///< Segédváltozó a konstruktor attribútumok számának
meghatározására
public:

    /// Konstruktor beállítja az attribútumokat
    /// @param o - ország megnevezése
    /// @param i - irányítószám megnevezése
    /// @param v - város megnevezése
    /// @param u - utca megnevezése
    /// @param h - házszám megnevezése
    Cim(const char *o, const char * i, const char *v, const char *u, const char * h)         :
orszag(o), iranyitoszam(i), varos(v), utca(u), hazszam(h)
    {
        kontruktor_t = 1;
    }

    /// Konstruktor beállítja az attribútumokat
    /// @param o - ország megnevezése
    Cim (const char *o)                          : orszag(o)
    {
        kontruktor_t = 2;
    }

    /// Konstruktor beállítja az attribútumokat
    /// @param i - irányítószám megnevezése
    /// @param v - város megnevezése
    Cim (const char *i, const char *v)            : iranyitoszam(i), varos(v)
    {
        kontruktor_t = 3;
    }
    /// Konstruktor beállítja az attribútumokat
    /// @param o - String típusú
    /// @param i - String típusú
    /// @param v - String típusú
    /// @param u - String típusú
    /// @param h - String típusú
    Cim(String o, String i,String v, String u, String h)         : orszag(o), iranyitoszam(i),
varos(v), utca(u), hazszam(h)
    {
        kontruktor_t = 1;
    }

    /// Teljes cím lekérdezése konstruktor paramétereinek száma alapján
    /// Minden adattag: teljes cím vesszővel elválasztva
```

```
    /// Csak országgal
    /// Csak Irányírószámmal és városnévvel
    /// @return - ha kontruktor_t == 1 akkor orszag + ", " + iranyitoszam + ", " + varos + ",
" + utca + " " + hazszam
    /// @return - ha kontruktor_t == 2 akkor orszag
    /// @return - ha kontruktor_t == 3 akkor iranyitoszam + ", "+  varos
    String getTeljesCim() const
    {
        if (kontruktor_t == 1)
        {
            return orszag + ", " + iranyitoszam + ", " + varos + ", " + utca + " " + hazszam;
        }
        if (kontruktor_t == 2)
        {
            return orszag;
        }
        else
        {
            return iranyitoszam + ", "+  varos;
        }

    }


    /// Attribútumok kiírása egy stream-re
    /// @param os - output stream referencia
    /// @return output stream referencia
    virtual std::ostream& print(std::ostream& os) const
    {
        return os << getTeljesCim() << std::endl;
    }

    /// Két cím egyezőségét vizsgálja
    /// @param c - jobb oldali operandus
    /// @return true, ha egyezik a két név
    bool operator==(const Cim& c) const
    {
        if (kontruktor_t == 1)
        {
            return orszag == c.orszag && iranyitoszam == c.iranyitoszam && varos == c.varos &&
utca == c.utca && hazszam == c.hazszam;
        }
        if (kontruktor_t == 2)
        {
            return orszag == c.orszag;
        }
        else
        {
            return iranyitoszam == c.iranyitoszam && varos == c.varos;
        }


    }
    /// Adatok beolvasása istreamről
    /// @return Cim - beolvasott adatokkal
    Cim beolvas_cim()
    {
        String o = "";
        String i = "";
        String v = "";
        String u = "";
        String h = "";
        std::cout << "Orszag: ";
        std::cin >> o;
        std::cout << "Iranyitoszam: ";
        std::cin >> i;
        std::cout << "Varos: ";
        std::cin >> v;
        std::cout << "Utca: ";
        std::cin >> u;
        std::cout << "Hazszam: ";
        std::cin >> h;
        return Cim(o,i,v, u, h);
    }

    /// Virtuális destruktor
    virtual ~Cim() {}
```

```
};

/// kiír az ostream-re (printCim)
/// @param os - ostream típusú objektum
/// @param s0 - Cim, amit kiírunk
/// @return os
std::ostream& operator<<(std::ostream& os, const Cim& s0);

#endif
```

6.3. gen.hpp

```
#include<iostream>

#include "memtrace.h"
#ifndef GEN_HPP
#define GEN_HPP


template<class T>
class GenTomb
{
protected:
    T *elems;       ///< T osztály pointer
    int size;       ///< elemek száma a tömbben
public:
    /// Alapértelmezett Konstruktor
    /// size - automatikusan 0 értéket állít be
    /// lefoglalja a helyet dinamikusan
    GenTomb(): size(0)
    {
        elems=new T[size];
    }
    /// Másoló konstruktor
    /// helyet dinamikusan foglalja
    /// @param s - konstans T-ket tartalmazó referenciát vesz át
    GenTomb(const GenTomb<T>& s)
    {
        size=s.size;
        elems=new T[size];
        for(int i=0; i<size; i++)
            elems[i]=s.elems[i];
    }
    /// Destruktor
    ~GenTomb()
    {
        delete[] elems;
    }
    /// Indexelõ operátor
    /// @return elems[inedx] - visszatér a tömb index-edik elemével.
    T& operator[](int index)
    {
        return elems[index];
    }
    /// Értékadó operátor
    GenTomb& operator=(T temp)
    {
        for(int i=0; i<size; i++)
            elems[i]=temp;
        return *this;
    }
    /// Függvény ami visszatér a tömb hosszával
    /// @return size
    int GetHossz()
    {
        return size;
    }

    /// Függvény ami hozzáad egy elemet a tömb végéhez
    /// @param n - Nevjegy pointer
    void push(Nevjegy* n)
    {
        T* temp = new T[size+1];
```

```
        for (int i = 0; i < size; i++)
        {
            temp[i] = elems[i];
        }
        temp[size] = n;
        size++;
        delete[] elems;
        elems = temp;

    }
    /// Függvény ami töröl egy elemet a tömbbõl
    /// Vizsgálja, hogy nem lehet üres a tömb
    /// @param n - Nevjegy pointer
    void pop(Nevjegy* n)
    {
        T* temp = new T[size-1];
        int j = 0;
        for (int i = 0; i < size; i++)
        {
            if (elems[i] != n && size != 0)
            {
                temp[j++] = elems[i];
            }
        }
        size--;
        delete[] elems;
        elems = temp;

    }


};
#endif
```

6.4. main.

6.5. nev.cpp

```
#include "nev.h"
#include "memtrace.h"
/// << operátor ami kiírja a Teljes nevet ostreamre
std::ostream& operator<<(std::ostream& os, const Nev& s0)
{
    os << s0.getTeljesnev();
    return os;
}
```

6.6. nev.h

```
#ifndef NEV_H
#define NEV_H

#include "string.h"
#include "memtrace.h"
class Nev
{
protected:
    String vezeteknev;      ///< Vezetéknév
    String keresztnev;      ///< Keresztnév
    String becenev;         ///< Becenév
    int kontruktor_t;       ///< Segédváltozó a konstruktor attribútumok számának
meghatározására
public:

    /// Konstruktor beállítja az attribútumokat
    /// @param k - keresztnév megnevezése
    /// @param v - vezetéknév megnevezése
    /// @param b - becenév megnevezése
    Nev(const char *v, const char *k, const char *b : vezeteknev(v), keresztnev(k), becenev(b)
    {
```

```
        kontruktor_t = 1;
    }

    /// Konstruktor beállítja az attribútumokat
    /// @param v - vezetéknév megnevezése
    /// @param k - keresztnév megnevezése
    Nev(const char *v, const char *k)                : vezeteknev(v), keresztnev(k)
    {
        kontruktor_t = 2;
    }

    /// Konstruktor beállítja az attribútumokat
    /// @param b - becenév megnevezése
    Nev(const char *b)                                : becenev(b)
    {
        kontruktor_t = 3;
    }
    /// Konstruktor beállítja az attribútumokat
    /// @param k - keresztnév megnevezése - String típusú
    /// @param v - vezetéknév megnevezése - String típusú
    /// @param b - becenév megnevezése - String típusú
    Nev(String v, String k, String b)          : vezeteknev(v), keresztnev(k), becenev(b)
    {
        kontruktor_t = 1;
    }

    /// Név lekérdezése értékadás alapján
    /// Mind a három adattagnak van értéke akkor visszatér a teljes névvel (közepén
becenévvel)
    /// Ha csak a vezetéknév és a keresztnév van megadva akkor csak azzal a kettővel tér
vissza
    /// És ha csak a becenevet adjuk meg akkor csak a becenévvel tér vissza
    /// @return - ha kontruktor_t == 1 akkor vezeteknev + " \"" + becenev + "\" " + keresztnev
    /// @return - ha kontruktor_t == 2 akkor vezeteknev + " " + keresztnev
    /// @return - ha kontruktor_t == 3 akkor becenev
    String getTeljesnev() const
    {

        if (kontruktor_t == 1)
        {
            return vezeteknev + " \"" + becenev + "\" " + keresztnev;
        }
        if (kontruktor_t == 2)
        {
            return vezeteknev + " " +keresztnev;
        }
        else
        {
            return becenev;
        }
    }

    /// Attribútumok kiírása egy stream-re
    /// @param os - output stream referencia
    /// @return output stream referencia
    virtual std::ostream& print(std::ostream& os) const
    {
        return os << getTeljesnev() << std::endl;
    }

    /// Két név egyezőségét vizsgálja konstrukor konstruktor paramétereinek száma alapján
    /// @param n - jobb oldali operandus
    /// @return true, ha egyezik a két név
    bool operator==(const Nev& n) const
    {
        if (kontruktor_t == 1)
        {
            return vezeteknev == n.vezeteknev && keresztnev == n.keresztnev && becenev ==
n.becenev;
        }
        if (kontruktor_t == 2)
        {
            return vezeteknev == n.vezeteknev && keresztnev == n.keresztnev;
        }
        else
        {
            return becenev == n.becenev;
```

```
        }

    }
    /// Adatok beolvasása istreamről
    /// @return Nev - beolvasott adatokkal
    Nev beolvas_nev()
    {
        String v = "";
        String k = "";
        String b = "";
        std::cout << "Vezetek: ";
        std::cin >> v;
        std::cout << "Kereszt: ";
        std::cin >> k;
        std::cout << "Bece: ";
        std::cin >> b;
        return Nev(v,k,b);
    }

    /// Virtuális destruktor
    virtual ~Nev() {}
};

/// kiír az ostream-re (printNev)
/// @param os - ostream típusú objektum
/// @param s0 - Nev, amit kiírunk
/// @return os
std::ostream& operator<<(std::ostream& os, const Nev& s0);


#endif
```

6.7. nevjegy.h

```
#ifndef NEVJEGY_H
#define NEVJEGY_H

#include "nev.h"
#include "string.h"
#include "cim.h"
#include "tel.h"
#include "memtrace.h"

class Nevjegy
{
private:
    Nev TeljesNev;      ///< Teljes név (Név konstruktortól függően teljes vagy csak a
becenév)
    Cim TeljesCim;      ///< Teljes cím (Cím konstruktortól függően az összes vagy csak az
ország, irányítószám, város)
    Tel TeljesTel;      ///< Teljes telefonszám (tel konstruktortól függően mindkettő vagy
csak a magán)
    String Email;
public:
    /// Konstruktor beállítja az attribútumokat
    /// @param n - nev
    /// @param c - cim
    /// @param t - telefonszam
    /// @param e - Email String
    Nevjegy(const Nev& n, const Cim& c, const Tel& t, String e) :TeljesNev(Nev(n)),
TeljesCim(Cim(c)), TeljesTel(Tel(t)), Email(String(e)) {}

    /// Függvény ami visszatér a TeljesNév értékkel
    /// @return TeljesNev
    Nev getNev()
    {
        return TeljesNev;
    }
    /// Függvény ami visszatér a TeljesCim értékkel
    /// @return TeljesCim
    Cim getCim()
    {
        return TeljesCim;
    }
```

```
    /// Függvény ami visszatér a TeljesTel értékkel
    /// @return TeljesTel
    Tel getTel()
    {
        return TeljesTel;
    }
    /// Virtuális függvény ami kiírja az adatokat "\n"-el elválasztva ostreamre
    /// @param os - ostream
    /// @return os << TeljesNev << "\n" << TeljesCim << "\n" << TeljesTel <<  "\n" << Email <<
"\n" << std::endl
    virtual std::ostream& print(std::ostream& os) const
    {
        return os << TeljesNev << "\n" << TeljesCim << "\n" << TeljesTel <<  "\n" << Email <<
"\n" << std::endl;
    }
    /// Virtuális függvény ami kiírja az csak a neveket
    /// @param os - ostream
    /// @return os << TeljesNev << "\n" << std::endl
    virtual std::ostream& print_nev(std::ostream& os) const
    {
        return os << TeljesNev << "\n" << std::endl;
    }


    /// Destruktor
    virtual ~Nevjegy() {}

};
#endif
```

6.8. string.cpp

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

/// Konstruktorok: egy char karakterből

String::String(char ch)
{
    len = 1;
    pData = new char[len+1];
    pData[0] = ch;
    pData[1] = '\0';
}

/// egy nullával lezárt char sorozatból

String::String(const char *p)
{
    len = strlen(p);
    pData = new char[len+1];
    strcpy(pData, p);
}

/// Másoló konstruktor: String-ből készít
String::String(const String& s1)
{
    len = s1.len;
    pData = new char[len+1];
    strcpy(pData, s1.pData);
}


/// Destruktor (disposeString)
String::~String()
{
    delete[] pData;
}


/// operator=
String& String::operator=(const String& rhs)
```

```
{
    if (this != &rhs)
    {
        delete[] pData;
        len = rhs.len;
        pData = new char[len+1];
        strcpy(pData, rhs.pData);
    }
    return *this;
}

/// + operátorok:
String String::operator+(const String& rhs) const
{
    String temp;
    temp.len = len + rhs.len;
    delete []temp.pData;
    temp.pData = new char[temp.len+1];
    strcpy(temp.pData, pData);
    strcat(temp.pData, rhs.pData);
    return temp;
}

/// << operator, ami kiír az ostream-re
std::ostream& operator<<(std::ostream& os, const String& s0)
{
    os << s0.c_str();
    return os;
}
/// Segéd strcmp függvény
int String::_strcmp(const String & rhs) const
{

    if (size() < rhs.size())
        return 1;
    else if (size() > rhs.size())
        return -1;

    return strcmp(c_str(), rhs.c_str());
}
/// == összehasonlító operátor
bool operator == (const String & lhs, const String & rhs)
{
    return lhs._strcmp(rhs) == 0;
}
/// >> operator, ami beolvas az istream-ről
std::istream& operator>>(std::istream& is, String& s0)
{
    unsigned char ch;
    s0 = String("");
    std::ios_base::fmtflags fl = is.flags();
    is.setf(std::ios_base::skipws);
    while (is >> ch)
    {
        is.unsetf(std::ios_base::skipws);
        if (isspace(ch))
        {
            is.putback(ch);
            break;
        }
        else
        {
            s0 = s0 + ch;
        }
    }
    is.setf(fl);
    return is;
}
```

6.9. string. h

```
#ifndef STRING_H
#define STRING_H
```

```
#include <iostream>
#include "memtrace.h"
class String
{
    char *pData;            ///< pointer
    size_t len;             ///< szó hossza lezáró nulla nélkül
public:
    /// Kiírunk egy tetszõleges szöveget.
    /// @param txt - nullával lezárt szövegre mutató pointer
    void printDbg(const char *txt = "") const
    {
        std::cout << txt << "[" << len << "], "
                  << (pData ? pData : "(NULL)") << '|' << std::endl;
    }

    /// Konstruktor:
    String() :pData(0), len(0) {}

    /// String hosszával történõ visszatérés
    /// @return len;
    size_t size() const
    {
        return len;
    }
    /// Stringet visszaadó függvény
    /// @return pData;
    const char* c_str() const
    {
        if (pData == NULL) return "";
        else return pData;
    }
    /// Destruktor
    virtual ~String();
    /// Konstruktor: egy char karakterbõl
    /// @param ch - karakter
    String(char ch);
    /// Konstruktor: egy nullával lezárt char sorozatból
    /// @param p - C típusú string
    String(const char *p);
    /// Másoló konstruktor
    /// @param s1 - String
    String(const String& s1);
    /// Értékadó operátor
    /// @param rhs - String
    String& operator=(const String& rhs);
    /// Összeadó operaátor - Két Stringet fûz össze
    /// @param rhs - String
    /// @return új String
    String operator+(const String& rhs) const ;
    /// Egyenlõség operátor - Két stringet hasonlít össze
    /// @param lhs - String
    /// @param rhs - String
    /// @return true - ha a két stirng ugyan az
    friend bool operator== (const String& lhs, const String& rhs);
    /// Saját strcmp függvény - segédfüggvény
    /// @param rhs - String
    /// @return 0 - ha a két string egyezik
    int _strcmp(const String & rhs) const;


};
/// Ostreamre író függvény
/// @param os - ostream típusú objektum
/// @param s0 - String, amit kiírunk
/// @return os
std::ostream& operator<<(std::ostream& os, const String& s0);
/// String operator+(char ch, const String& str);
/// @param ch - karakter
/// @param str - String
/// @return új String
inline String operator+(char ch, const String& str)
{
    return String(ch) + str;
}
///Istreamrõl beolvasó függvény
/// @param is - istream típusú objektum
/// @param s0 - String, amibe beolvas
```

```
/// @return is
std::istream& operator>>(std::istream& is, String& s0);

#endif
```

6.10. tel.cpp

```
#include "tel.h"
#include "memtrace.h"
/// << operátor ami kírja a Teljes telefonszámot ostreamre
std::ostream& operator<<(std::ostream& os, const Tel& s0)
{
    os << s0.getTel();
    return os;
}
```

6.11. tel.h

```
#ifndef TEL_H
#define TEL_H

#include "string.h"
#include "memtrace.h"
class Tel
{
protected:
    String munkahelyi;              ///< Munkahelyi telefonszám
    String magan;                   ///< Magán telefonszám
    bool kontruktor_t;              ///< Segédváltozó a konstruktor attribútumok számának
meghatározására
public:
    /// Konstruktor beállítja az attribútumokat
    /// @param munk - munkahelyi telefonszám megnevezése
    /// @param mag - magán telefonszám megnevezése
    Tel(const char *munk, const char *mag)          :munkahelyi(munk), magan(mag)
    {
        kontruktor_t = true;
    }
    /// Konstruktor beállítja az attribútumokat
    /// @param mag - magán telefonszám megnevezése
    Tel(const char *mag)                            :magan(mag)
    {
        kontruktor_t = false;
    }
    /// Konstruktor beállítja az attribútumokat
    /// @param munk - String típusú
    /// @param mag - String típusú
    Tel(String munk, String mag)          :munkahelyi(munk), magan(mag)
    {
        kontruktor_t = true;
    }


    /// Telefonszám lekérdezése értékadás alapján
    /// Munkahelyi és magán telefonszámmal való visszatérés
    /// Illetve csak magán telefonszámmal való visszatérés
    /// @return - ha kontruktor_t == true akkor munkahelyi + " / " + magan;
    /// @return - ha kontruktor_t == false akkor magan
    String getTel() const
    {
        if (kontruktor_t)
        {
            return munkahelyi + " / " + magan;
        }
        else
        {
            return magan;
        }
    }

    /// Attribútumok kiírása egy stream-re
    /// @param os - output stream referencia
```

```
    /// @return output stream referencia
    virtual std::ostream& print(std::ostream& os) const
    {
        return os << getTel() << std::endl;
    }

    /// Két Telefonszám egyezőségét vizsgálja konstruktor paramétereinek száma alapján
    /// @param t - jobb oldali operandus
    /// @return true, ha egyezik a két telefonszám
    bool operator==(const Tel& t) const
    {
        if (kontruktor_t)
        {
            return munkahelyi == t.munkahelyi && magan == t.magan;
        }
        else
        {
            return magan == t.magan;
        }

    }
    /// Adatok beolvasása istreamről
    /// @return Tel - beolvasott adatokkal
    Tel beolvas_tel()
    {
        String munk = "";
        String mag = "";
        std::cout << "Munkahelyi: ";
        std::cin >> munk;
        std::cout << "Magan: ";
        std::cin >> mag;

        return Tel(munk, mag);
    }

    /// Virtuális destruktor
    virtual ~Tel() {}
};
/// kiír az ostream-re (printTel)
/// @param os - ostream típusú objektum
/// @param s0 - Tel, amit kiírunk
/// @return os
std::ostream& operator<<(std::ostream& os, const Tel& s0) ;

#endif
```

6.12. telefonkonyv.cpp

```
#include "telefonkonyv.h"
#include <iostream>
#include "memtrace.h"
using std::cout;
using std::endl;
using std::cin;

/// Függvény ami hozzáadja a tömbhöz az adott névjegyet
void Telefonkonyv::Hozzaad (Nevjegy* n)
{
    ember.push(n);
}
/// Függvény ami kitörli az adott névjegyet a tömbbõl
void Telefonkonyv::Tororl (Nevjegy* n)
{

    if (ember.GetHossz() == 0)
    {
        cout << "ERROR" << endl;
    }
    else
    {
        ember.pop(n);
    }

}
```

```
/// Függvény ami kilistázza az adatokat
void Telefonkonyv::listaz (std::ostream& os)
{
    if (ember.GetHossz() == 0)
    {
        cout << "Ures a tomb!" << endl;
    }
    for(int i = 0; i < ember.GetHossz(); i++)
    {
        os << i+1 << ". nevjegy:" << std::endl;
        ember[i]->print(os);
    }
}
/// Függvény ami kilistázza csak a neveket
void Telefonkonyv::listaz_nev (std::ostream& os)
{
    if (ember.GetHossz() == 0)
    {
        cout << "Ures a tomb!" << endl;
    }
    for(int i = 0; i < ember.GetHossz(); i++)
    {
        os << i+1 << ". nevjegy:" << std::endl;
        ember[i]->print_nev(os);
    }
}
/// Függvény ami megkeresi az adott nevû névjegyet
void Telefonkonyv::KeresNev(const Nev& n, std::ostream& os)
{
    int db = 0;
    if (ember.GetHossz() == 1)
    {
        if (n == ember[0]->getNev())
        {
            ember[0]->print(os);
            db++;
        }
    }
    else
    {
        for(int i = 0; i < ember.GetHossz(); i++)
        {
            if (n == ember[i]->getNev())
            {
                ember[i]->print(os);
                db++;
            }
        }
    }
    if (db == 0)
    {
        cout << "Nincs ilyen nev az adatbazisban!" << endl;
    }
}
/// Függvény ami megkeresi az adott telefonszámú névjegyet
void Telefonkonyv::KeresTel(const Tel& n, std::ostream& os)
{
    int db = 0;
    if (ember.GetHossz() == 1)
    {
        if (n == ember[0]->getTel())
        {
            ember[0]->print(os);
            db++;
        }
    }
    else
    {
        for(int i = 0; i < ember.GetHossz(); i++)
        {
            if (n == ember[i]->getTel())
            {
                ember[i]->print(os);
                db++;
            }
        }
    }
```

```
    if (db == 0)
    {
        cout << "Nincs ilyen telefonszam az adatbazisban!" << endl;
    }
}
/// Függvény ami megkeresi az adott címû névjegyet
void Telefonkonyv::KeresCim(const Cim& n, std::ostream& os)
{
    int db = 0;
    if (ember.GetHossz() == 1)
    {
        if (n == ember[0]->getCim())
        {
            ember[0]->print(os);
            db++;
        }
    }
    else
    {
        for(int i = 0; i < ember.GetHossz(); i++)
        {
            if (n == ember[i]->getCim())
            {
                ember[i]->print(os);
                db++;
            }
        }
    }
    if (db == 0)
    {
        cout << "Nincs ilyen cim az adatbazisban!" << endl;
    }
}
```

6.13. telefonkonyv.h

```
#ifndef TELELFONKONYV_H
#define TELELFONKONYV_H
#include <iostream>
#include "string.h"
#include "nevjegy.h"
#include "gen.hpp"
#include "memtrace.h"

class Telefonkonyv
{
    GenTomb<Nevjegy*> ember;        ///< névjegy adatai ami egy generikus tömbben van
eltárolva
public:
    /// Függvény ami meghívja a push függvényt és hozzáad egy névjegy-et a tömbhöz
    /// @param n - Névjegy típusú pointer
    void Hozzaad (Nevjegy* n);
    /// Függvény ami meghívja a pop függvényt és töröl egy névjegy-et a tömbből
    /// @param n - Névjegy típusú pointer
    void Tororl (Nevjegy* n);
    /// Függggvény ami ostreamre kiírja az összes adatot ami a tömbben található
    /// @param os - ostream
    void listaz (std::ostream& os);
    /// Függvény ami ostreamre kiírja az összes embernek csak a nevét
    /// @param os - ostream
    void listaz_nev (std::ostream& os);
    /// Függvény ami megmondja, hogy hány névjegy található a tömbben
    /// @return ember.GetHossz - GenTömb hosszával
    int darab()
    {
        return ember.GetHossz();
    }

    /// Függvény ami kikeresi a tömbben az adott nevet és kiírja ostreamre
    /// A függvény figyelembe veszi hogy hány adat van megadva, így tehát képesek vagyunk csak
becenév vagy kereszt és vezetéknév alapján keresni.
```

```
    /// @param n - Név típus
    /// @param os - ostream
    void KeresNev(const Nev& n, std::ostream& os);
    /// Függvény ami kikeresi a tömbben az adott telefonszámot és kiírja ostreamre
    /// A függvény figyelembe veszi hogy hány adat van megadva, így tehát képesek vagyunk csak
magán telefonszám alapján keresni.
    /// @param t - Tel típus
    /// @param os - ostream
    void KeresTel(const Tel& t, std::ostream& os);
    /// Függvény ami kikeresi a tömbben az adott címet és kiírja ostreamre
    /// A függvény figyelembe veszi hogy hány adat van megadva, így tehát képesek vagyunk csak
ország alapján keresni. Továbbá képesek vagyunk csak irányitószám és város megadásával keresni
is.
    /// @param c - Cim típus
    /// @param os - ostream
    void KeresCim(const Cim& c, std::ostream& os);
};
#endif
```

6.14. Main

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

using std::cout;
using std::endl;
using std::cin;

#include "memtrace.h"

void test_1()
{
    cout << "************************* Teszt 1 - Adatok felvetele kodban
*************************" << "" << endl;


    cout << "\nAdatok felvetele kodban \n" << endl;
    Telefonkonyv t1;
    Nevjegy
n1(Nev("Vezeteknev","Keresztnev","Becenev"),Cim("Orszag","Irany","Varos","Utca/ut","Hazszam"),
Tel("1234567890","9876543210"), "email@email.com");
    t1.Hozzaad(&n1);
    Nevjegy n2(Nev("Teszt","Elek","TE"),Cim("Magyarorszag","1111","Budapest","Alma
utca","10"), Tel("06301234456","06306555002"), "elek@gmail.com");
    t1.Hozzaad(&n2);
    Nevjegy n3(Nev("Ferenci", "Viva", "Viva"),Cim("Magyarorszag","2234","Maglod","Sip
utca","74"), Tel("06295760940","06448599209"), "ferenci@viva.com");
    t1.Hozzaad(&n3);
    Nevjegy n4(Nev("asd", "asd", "asd"),Cim("asd","asd","asd","asd asd","asd"),
Tel("asd","asd"), "asd@asd.com");
    t1.Hozzaad(&n4);

    int db = t1.darab();
    cout<<"SIM kartyan tarolt nevjegyek: "<< db <<"\n"<<endl;
    t1.listaz(cout);

    cout<< n2.getNev() << " nevu szemely torlese "<<endl;
    t1.Tororl(&n2);
    t1.listaz_nev(cout);

    t1.Tororl(&n3);
    t1.Tororl(&n4);
    t1.Tororl(&n1);

}

void test_2()
{
    cout << "\n************************* Teszt 2 - Adatok felvetele szabvanyos bemenetrol
*************************" << "" << endl;

    Telefonkonyv t1;
    cout << "Adatok felvetele szabvanyos bemenetrol\n" << endl;
```

```
    cout << "Nevjegy felvetele: " << endl;

    Nev n5 = n5.beolvas_nev();
    Cim c5 = c5.beolvas_cim();
    Tel t5 = t5.beolvas_tel();
    cout << "Email: ";
    String e5;
    cin >> e5;

    Nevjegy nevjegy_beolvas(n5, c5, t5, e5);
    t1.Hozzaad(&nevjegy_beolvas);

    cout <<""<<endl;
    nevjegy_beolvas.print(cout);
    t1.Tororl(&nevjegy_beolvas);
}

void test_3()
{
    cout << "\n ************************* Teszt 3 - Kereses *************************" << ""
<< endl;

    Telefonkonyv t1;
    Nevjegy
n1(Nev("Vezeteknev","Keresztnev","Becenev"),Cim("Orszag","Irany","Varos","Utca/ut","Hazszam"),
Tel("1234567890","9876543210"), "email@email.com");
    t1.Hozzaad(&n1);
    Nevjegy n2(Nev("Teszt","Elek","TE"),Cim("Magyarorszag","1111","Budapest","Alma
utca","10"), Tel("06301234456","06306555002"), "elek@gmail.com");
    t1.Hozzaad(&n2);
    Nevjegy n3(Nev("Ferenci", "Viva", "Viva"),Cim("Magyarorszag","2234","Maglod","Sip
utca","74"), Tel("06295760940","06448599209"), "ferenci@viva.com");
    t1.Hozzaad(&n3);

    cout << "##################\nAdatok keresese Nev alapjan  \n" << endl;

    Nev nev1("Vezeteknev","Keresztnev", "Becenev");
    printf("-- Ezt keressuk: ");
    nev1.print(cout);
    t1.KeresNev(nev1, cout);

    Nev nev2("Teszt", "Elek");
    printf("-- Ezt keressuk: ");
    nev2.print(cout);
    t1.KeresNev(nev2, cout);

    Nev nev3("Viva");
    printf("-- Ezt keressuk: ");
    nev3.print(cout);
    t1.KeresNev(nev3, cout);

    cout << "##################\nAdatok keresese Telefonszam alapjan   \n" << endl;

    Tel tel1("06301234456","06306555002");                          //mind2 alapján
    printf("-- Ezt keressuk: ");
    tel1.print(cout);
    t1.KeresTel(tel1, cout);

    Tel tel2("06448599209");                                         // csak magán telefonszám
alapján
    printf("-- Ezt keressuk: ");
    tel2.print(cout);
    t1.KeresTel(tel2, cout);

    cout << "##################\nAdatok keresese Cim alapjan  \n" << endl;

    Cim cim1("Orszag","Irany","Varos","Utca/ut","Hazszam");
    printf("-- Ezt keressuk: ");
    cim1.print(cout);
    t1.KeresCim(cim1, cout);

    Cim cim2("Magyarorszag");
    printf("-- Ezt keressuk: ");
    cim2.print(cout);
    t1.KeresCim(cim2, cout);

    Cim cim3("2234","Maglod");
```

```
    printf("-- Ezt keressuk: ");
    cim3.print(cout);
    t1.KeresCim(cim3, cout);

    t1.Tororl(&n1);
    t1.Tororl(&n2);
    t1.Tororl(&n3);

}

void test_4()
{
    cout << "\n ************************* Teszt 4 - Hibas adatok *************************"
<< "" << endl;

    Telefonkonyv t1;
    cout << "Adat torlese egy ures tombbol: " << endl;
    Nevjegy n6(Nev("Teszt","Elek","TE"),Cim("Magyarorszag","1111","Budapest","Alma
utca","10"), Tel("06301234456","06306555002"), "elek@gmail.com");
    t1.Tororl(&n6);

    cout << "Olyan adat keresese ami nincs benne a tombben: " << endl;

    Cim cim4("Nincs","Nincs");
    printf("-- Ezt keressuk: ");
    cim4.print(cout);
    t1.KeresCim(cim4, cout);



}

int main()
{
    cout << "#########################################################" <<endl;
    cout << "#                                                       #" <<endl;
    cout << "#               TELEFONKONYV ALKALMAZAS                 #" <<endl;
    cout << "#                         v1.0                          #" <<endl;
    cout << "#                                                       #" <<endl;
    cout << "#########################################################\n\n" <<endl;

    test_1();
    test_2();
    test_3();
    test_4();

    return 0;
}
```