Házi feladat

Programozás alapjai 2.

NHF4 - Kész

Kussa Richárd

RONAOF

2023. május 28.

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
- cím
- munkahelyi szám
- privát szám

Az alkalmazással minimum a következő műveleteket kívánjuk elvégezni:

- adatok felvétele
- adatok törlése
- listázás

A rendszer lehet bővebb funkcionalitású (pl. módosítás, keresés), 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!

2. Feladatspecifikáció

A feladat egy nyilvántartás elkészítése. Mivel a tárolt adatokat később még bővíteni szeretnénk, ezért érdemes lehet egy heterogén gyűjtemény felállítása objektummodellünkben.

Fontos, hogy legyen tároló perzisztens. Az adatok felvételét automatikus fájlba mentés kell kövesse. Értelemszerűen ezeket az adatokat fájlból visszatölteni is képes kell legyen a program.

Az adatok felvétele, törlése és kilistázása mellett az adatokat másolni is tudni kell. Az adattárolásnak pedig mindenkoron dinamikusan kell végbe mennie.

Mindemellett szükségünk lesz generikus keresésre, már csak azért is, mert nem szükséges, hogy kétszer pontosan ugyanaz az adat szerepeljen. Vagyis az újnak gondolt adatok felvételekor mindig ellenőrizni kell, hogy vajon telefonkönyvünkben szerepelnek-e már.

3. Pontosított feladatspecifikáció

A feladat egy nyilvántartás elkészítése heterogén kollekcióval. Készíteni kell tehát néhány különböző adatokat tároló leszármazott osztályokat a kontakt osztályához.

Ilyenek lehetnek például:

- barát (becenévvel)
- munkatárs (beosztással)
- családtag (születésnappal)

A felsoroltakon kívül más leszármazott osztály is szóba jöhet, a cél csak az, hogy legalább kettő legyen megvalósítva.

A következő függvényeknek pedig működniük kell az egész telefonkönyvre:

- kiírás
- beolvasás
- értékadás
- másolás

Fontos, hogy legyen tároló perzisztens. Az adatok felvételét automatikus fájlba mentés kell kövesse. Értelemszerűen ezeket az adatokat fájlból visszatölteni is képes kell legyen a program.

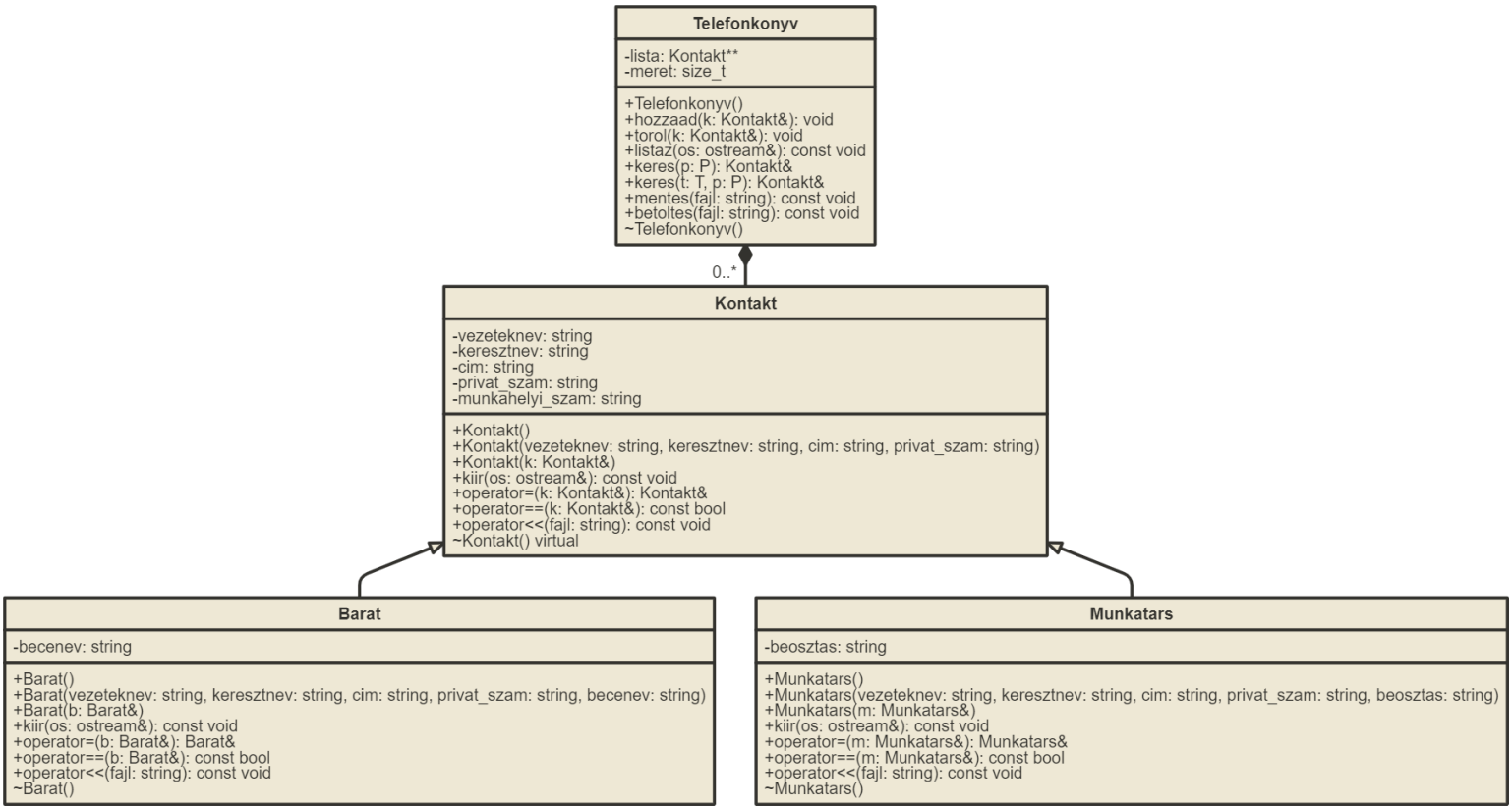
Az adattárolás mindenkor dinamikusan kell végbe menjen.

Mindemellett szükségünk lesz generikus keresésre, már csak azért is, mert nem szükséges, hogy kétszer pontosan ugyanaz az adat szerepeljen. Vagyis az újnak gondolt adatok felvételekor mindig ellenőrizni kell, hogy vajon telefonkönyvünkben szerepelnek-e már.

4. Terv

4.1. Objektum terv

Alább látható az objektummodell. A *Kontakt* osztályból több osztály is származhat, mint ami itt látható, de legalább ennyi kell legyen.



A *Kontakt* objektumokat egy pointer lista fogja tárolni a *Telefonkonyv* osztályban. Ehhez szükség lesz a *hozzaad* függvényre, amely a paraméterként kapott *Kontakt* pointert felveszi a listába. Ennek párja pedig a *torol* függvény, amely a paraméterként kapott *Kontakt*-ot törli a listából, feltéve, hogy az benne van.

A *listaz* függvény feladata, hogy a lista minden elemét kiírja a paraméterként kapott kimenetre.

A generikus keresést a *keres* függvények valósítják meg. Mindkettő sablonparaméterként vesz át egy komparátor függvényt, de az egyik egy másik sablonparamétert is kap, amire bizonyos komparátor függvényeknek szükséges.

A *mentes* függvény végzi a fájlba mentést, ennek párja a *betoltes* pedig visszatölteni tudja a listát. Mindkettőnek szüksége lesz egy fájlnévre paraméternek.

A Kontakt és a leszármazott osztályok mind rendelkeznek *kiir* függvénnyel és *értékadó operátorral,* valamint *összehasonlító operátorral* is.

4.2 Algoritmusok

4.2.1 Lista szerkesztése

A *hozzaad* és *torol* függvényeknek tudniuk kell ellenőrizni, hogy az adott *Kontakt* szerepel-e már a listában. Abszolút szükségtelen két teljesen megegyező adatot egyszerre szerepeltetni egy telefonkönyvben és törölni sem tudunk olyan adatot, ami eleve benne se volt a tárolónkban.

Fontos, hogy mindkettő dinamikusan működjön.

hozzaad(k: Kontakt&) void:
ciklus i = 0 - tól *meret* - 1 -ig
    if *k.osztaly* = *lista*[i].*osztály*
        if *k* == *lista*[i] // összehasonlító operátorral
            dobjon kivételt
…

A *torol* a fentihez hasonlóan megkeresi azt az elemet, amelyik paraméterként adott, de az csak akkor dob kivételt, ha nem találja azt.

4.2.2 Objektumok listázása

A *listaz* függvény csupán végigfut egyszer a listán és minden elemnek meghívja annak saját *kiir* tagfüggvényét.

4.2.3 Keresés a listában

A generikus kereséshez két *keres* függvényt lehet írni, mert predikátum függvényből is kétféle lehet:

- Olyan, amely egy „leg” elemet keres (pl.: névsorban első, -utolsó, leghosszabb nevű stb.)
- Olyan, amely valamilyen adategyezést keres (pl.: privát száma: 7777777)

Ez utóbbihoz ezért már egy második sablonparaméter is kell.

keres(p: P):
visszaad *p*(*lista*)
// hogy egy „leg” elemet megtaláljunk, ahhoz az egész listát oda kell majd adni

keres(t: T, p: P):
ciklus i = 0 - tól *meret* - 1 -ig
    if *p*(*t, lista*[i])
        visszaad *lista*[i]

A komparátor függvények sokfélék lehetnek, külön nincsenek is feltüntetve az objektummodellben, már csak azért sem, mert akár a fentebb látható osztályokon kívül is elhelyezkedhetnek.

4.2.4 Fájlkezelés

Korábban már említettük, hogy ideális lenne egy automatikus mentés funkció, amely például egy áramkimaradás esetén az adatvesztés lehetőségét minimalizálná, erről szólna a *mentes* függvény.

mentes(fajlnev: char*):
megnyit(*fajlnev*)
ciklus i = 0 - tól *meret* - 1 -ig
    fájlba ír ← *lista*[i] *Kontakt* adatai szóközzel elválasztva
    if *osztaly* != kontakt
        if *osztaly* = barat
            fájlba ír ← *lista*[i] *Barat* adatai
        else if osztaly = munkatars
            fájlba ír ← *lista*[i] *Munkatars* adatai
        else if osztaly = …
    új sor
bezár(*fajlnev*)

Ez attól lesz automatikus, hogy minden új *Kontakt* listába való felvétele után a *hozzaad* függvény meghívja majd (persze ez azt is jelenti, hogy amikor újraindítjuk a programot és létrehozunk egy új telefonkönyvet új kontaktokkal, akkor az alapértelmezetten adott fájlban az addig elmentett adatok felülíródnak, vagyis elvesznek, szóval ha azt akarjuk, hogy telefonkönyvünk megmaradjon érdemes egyszer meghívni ezt a függvényt egy egyedi fájlnévvel is).

A *betolt* függvény dolga, hogy az elmentett adatokat fájlból visszatöltse. Ez már nem automatikus, külön meg kell hívni miután létrehoztunk egy telefonkönyvet. Algoritmusa nem sokkal bonyolultabb az előzőnél. Az információk sorrendjét ismerjük, és az első adat, azaz az *osztaly* alapján azt is tudjuk, hogy mennyi és milyen adatot kell felvenni. Az *mentes*-hez hasonló if – else if szerkezettel vagy case szerkezettel tudjuk ezt megoldani. A függvény osztály alapján meghívja a megfelelő konstruktorokat.

4.2.5 Tesztprogram algoritmusai

A tesztprogram a standard inputról file végéig olvas. Az első beolvasott adat egy teszteset sorszámot jelent. Ezt egy megjegyzés lehet az adott sorban. A beolvasott szám dönti el, hogy melyik teszteset fut a megjegyzés pedig az adott tesztesetre vonatkozhat.

A teszteseteknek a következő ellenőrzéseket végzik:

- Beolvasás/értékadás/másolás helyessége
- Törlés megfelelő működése
- Keresés helyessége
- Fájlkezelés helyes működése

Emellett a dinamikus memóriakezelés hiányát vagy hibáját is vizsgálni kell és jelezni, amennyiben az fellép.

5. Megvalósítás

A feladat megoldása egy ősosztály, két származtatott osztály, valamint egy ezen osztályok objektumait tárolni tudó osztály elkészítését igényelte. A tervezéshez képest néhol változott az osztályok interfésze, mert ezen esetekben volt logikusabb megoldás a korábbiakban vázoltakhoz képest. Ezenkívül a tesztelés algoritmusai is módosultak.

A korábban *telefonkonyv.hpp* fájlban lévő osztályok szét lettek szedve több fájlba, így ezek a *telefonkonyv.h, telefonkonyv.cpp, kontakt.h és kontakt.cpp* fájlokba kerültek. A tesztprogram maradt a *telefonkonyv_main.cpp* fájlban, de lett egy *szovegek.h* fájl is, amiben a program különböző hiba- és egyéb üzenetei vannak definiálva. A továbbiakban bemutatom a fontosabb interfészeket és algoritmusokat a program forrása alapján generált dokumentáció felhasználásával.

5.1. Az ősosztály és az alosztályok bemutatása

5.1.1 A Kontakt ősosztály

Az osztálynak std::string típusú védett (protected) adatai vannak, hogy az alosztályok felől elérhetőek legyenek. A legtöbb implicit függvénye megfelően működik:

- másoló konstruktor
- értékadó operátor

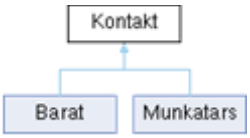
De konstruktorból olyat használunk, amiben paraméterként meg lehet adni az adatokat, és az implicit destruktort már nem használhattuk, mivel mi Kontakt pointereket tárolunk és a származtatott osztályoknak eltérő az adattípusa, ezért ez a destruktor virtuális kellett legyen.

Minden adathoz született get függvény is (get_vez, get_ker stb.), hogy azok a tároló felől is lekérdezhetőek legyenek.

Újdonság még a virtuális *clone* függvény, amely egy másolatot készít az objektumról és azt pointerként küldi vissza. Ez a függvény teljes tárolók értékadásánál játszik nagy szerepet (deep copy).

Meg kell említeni még a *string_keres* függvényt, amely a tárolt adatokban keresi a megadott szövegrészletet (*adat* -ot a *tipus* -ban, ezt később még részletezzük), és ha egyezést talál true -t ad vissza.

Kontakt Class Reference



Public Member Functions

Kontakt (string v="", string k="", string c="", string m="", string p="")
Konstruktor.

string **get_vez** ()
Adatok lekérdezése.

string **get_ker** ()
string **get_cim** ()
string **get_mun** ()
string **get_pri** ()
virtual void **kiir** (ostream &os=cout)
virtual **Kontakt** * **clone** ()
Másolat készítése (deep copy).

virtual bool **operator==** (**Kontakt** *k)
virtual **~Kontakt** ()
Destruktor.

virtual bool **string_keres** (string adat, string tipus)

Protected Attributes

string **vezeteknev**
string **keresztnev**
string **cim**
string **munkahelyi_szam**
string **privat_szam**

Detailed Description

Definition at line **31** of file **kontakt.h**.

Constructor & Destructor Documentation

Kontakt::Kontakt (string *v* = "", string *k* = "", string *c* = "", string *m* = "", string *p* = "")[inline]

Konstruktor.

Definition at line **40** of file **kontakt.h**.

```
00044                            : vezeteknev(v), keresztnev(k), cim(c),
munkahelyi_szam(m), privat_szam(p) {}
```

virtual Kontakt::~Kontakt () [inline], [virtual]

Destruktor.

Definition at line **65** of file **kontakt.h**.

```
00065 {}
```

Member Function Documentation

Kontakt * Kontakt::clone () [virtual]

Másolat készítése (deep copy).

Reimplemented in **Barat** (*p.15*), and **Munkatars** (*p.18*).

Definition at line **91** of file **kontakt.cpp**.

```
00091 { return new Kontakt(*this); }
```

string Kontakt::get_cim ()

Definition at line **15** of file **kontakt.cpp**.

```
00015 { return cim; }
```

string Kontakt::get_ker ()

Definition at line **14** of file **kontakt.cpp**.

```
00014 { return keresztnev; }
```

string Kontakt::get_mun ()

Definition at line **16** of file **kontakt.cpp**.

```
00016 { return munkahelyi_szam; }
```

string Kontakt::get_pri ()

Definition at line **17** of file **kontakt.cpp**.

```
00017 { return privat_szam; }
```

string Kontakt::get_vez ()

Adatok lekérdezése.

Definition at line **13** of file **kontakt.cpp**.

```
00013 { return vezeteknev; }
```

void Kontakt::kiir (ostream & *os* = cout) [virtual]

Egy kontakt adatainak kiírása.

Parameters

os	- output stream, default: cout

Reimplemented in **Barat** (*p.16*), and **Munkatars** (*p.19*).

Definition at line **21** of file **kontakt.cpp**.

```
00021                                {
00022     os << "Kontakt;";
00023     (vezeteknev != "") ? os << vezeteknev << ";" : os << "nincs;";
00024     (keresztnev != "") ? os << keresztnev << ";" : os << "nincs;";
00025     (cim != "") ? os << cim << ";" : os << "nincs;";
00026     (munkahelyi_szam != "") ? os << munkahelyi_szam << ";" : os << "nincs;";
00027     (privat_szam != "") ? os << privat_szam : os << "nincs";
00028     os << endl;
00029 }
```

bool Kontakt::operator== (Kontakt * *k*) [virtual]

Értékek összehasonlítása.

Parameters

k	- kontakt pointere

Reimplemented in **Barat** (*p.16*), and **Munkatars** (*p.19*).

Definition at line **58** of file **kontakt.cpp**.

```
00058                                     {
00059     if(vezeteknev != k->vezeteknev) return false;
00060     if(keresztnev != k->keresztnev) return false;
00061     if(cim != k->cim) return false;
00062     if(munkahelyi_szam != k->munkahelyi_szam) return false;
00063     if(privat_szam != k->privat_szam) return false;
00064     return true;
00065 }
```

bool Kontakt::string_keres (string *adat*, string *tipus*) [virtual]

Szöveg összehasonlító függvény, ami igazat ad vissza, ha az adat része a tipus által jelölt adatnak.

Parameters

adat	- szöveg
tipus	- ahol keressük (vez, ker, cim, mun, pri, bec, beo)

Reimplemented in **Barat** (*p.16*), and **Munkatars** (*p.19*).

Definition at line **95** of file **kontakt.cpp**.

```
00095                                                    {
00096     if(tipus == "vez") return vezeteknev.find(adat) != string::npos;
00097     else if(tipus == "ker") return keresztnev.find(adat) != string::npos;
00098     else if(tipus == "cim") return cim.find(adat) != string::npos;
00099     else if(tipus == "mun") return munkahelyi_szam.find(adat) != string::npos;
00100     else if(tipus == "pri") return privat_szam.find(adat) != string::npos;
00101     else {
00102         if(vezeteknev.find(adat) != string::npos ||
00103            keresztnev.find(adat) != string::npos ||
00104            cim.find(adat) != string::npos ||
00105            munkahelyi_szam.find(adat) != string::npos ||
00106            privat_szam.find(adat) != string::npos) { return true; }
00107     }
00108     return false;
00109 }
```

Member Data Documentation

string Kontakt::cim [protected]

Definition at line **35** of file **kontakt.h**.

string Kontakt::keresztnev [protected]

Definition at line **34** of file **kontakt.h**.

string Kontakt::munkahelyi_szam `[protected]`

Definition at line **36** of file **kontakt.h**.

string Kontakt::privat_szam `[protected]`

Definition at line **37** of file **kontakt.h**.

string Kontakt::vezeteknev `[protected]`

Definition at line **33** of file **kontakt.h**.

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

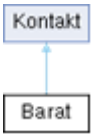
kontakt.hkontakt.cpp

5.1.2 A Barat és Munkatars származtatott osztályok

Tagfüggvényeik csak kis mértékben térnek el az ősosztályétól, a legtöbb esetben csak override történik, de például az impilicit destruktorok itt már megfelelően működnek, ezért azokat használjuk.

Barat Class Reference

Inheritance diagram for Barat:



Public Member Functions

Barat (string v="", string k="", string c="", string m="", string p="", string b="")
Konstruktor.

string **get_bec** ()
Speciális adat lekérdezése.

void **kiir** (ostream &os=cout)
Barat * **clone** ()
Másolat készítése (deep copy).

bool **operator==** (**Kontakt** *k)
bool **string_keres** (string adat, string tipus)

Public Member Functions inherited from Kontakt

Kontakt (string v="", string k="", string c="", string m="", string p="")
Konstruktor.

string **get_vez** ()
Adatok lekérdezése.

string **get_ker** ()
string **get_cim** ()
string **get_mun** ()
string **get_pri** ()
virtual void **kiir** (ostream &os=cout)
virtual **Kontakt** * **clone** ()
Másolat készítése (deep copy).

virtual bool **operator==** (**Kontakt** *k)
virtual **~Kontakt** ()
Destruktor.

virtual bool **string_keres** (string adat, string tipus)

Additional Inherited Members

Protected Attributes inherited from Kontakt

string **vezeteknev**
string **keresztnev**
string **cim**
string **munkahelyi_szam**
string **privat_szam**

Detailed Description

Definition at line **78** of file **kontakt.h**.

Constructor & Destructor Documentation

Barat::Barat (string *v* = "", string *k* = "", string *c* = "", string *m* = "", string *p* = "", string *b* = "") [inline]

Konstruktor.

Definition at line **82** of file **kontakt.h**.

```
00087                           : Kontakt(v, k, c, p, m), becenev(b) {}
```

Member Function Documentation

Barat * Barat::clone () [virtual]

Másolat készítése (deep copy).

Reimplemented from **Kontakt** (*p.12*).

Definition at line **92** of file **kontakt.cpp**.

```
00092 { return new Barat(*this); }
```

string Barat::get_bec ()

Speciális adat lekérdezése.

Definition at line **18** of file **kontakt.cpp**.

```
00018 { return becenev; }
```

void Barat::kiir (ostream & *os* = cout) [virtual]

Egy barát adatainak kiírása.

Parameters

os	- output stream, default: cout

Reimplemented from **Kontakt** (*p.12*).

Definition at line **31** of file **kontakt.cpp**.

```
00031                                {
00032     os << "Barat;";
00033     (get_vez() != "") ? os << get_vez() << ";" : os << "nincs;";
00034     (get_ker() != "") ? os << get_ker() << ";" : os << "nincs;";
00035     (get_cim() != "") ? os << get_cim() << ";" : os << "nincs;";
00036     (get_mun() != "") ? os << get_mun() << ";" : os << "nincs;";
00037     (get_pri() != "") ? os << get_pri() << ";" : os << "nincs;";
00038     (becenev != "") ? os << becenev : os << "nincs";
00039     os << endl;
00040 }
```

bool Barat::operator== (Kontakt * *k*) [virtual]

Értékek összehasonlítása.

Parameters

k	- kontakt pointere

Reimplemented from **Kontakt** (*p.13*).

Definition at line **67** of file **kontakt.cpp**.

```
00067                                    {
00068     Barat* l = dynamic_cast<Barat*> (k);
00069     if(l == NULL) return false;
00070     if(vezeteknev != l->vezeteknev) return false;
00071     if(keresztnev != l->keresztnev) return false;
00072     if(cim != l->cim) return false;
00073     if(munkahelyi_szam != l->munkahelyi_szam) return false;
00074     if(privat_szam != l->privat_szam) return false;
00075     if(becenev != l->becenev) return false;
00076     return true;
00077 }
```

bool Barat::string_keres (string *adat*, string *tipus*) [virtual]

Szöveg összehasonlító függvény, ami igazat ad vissza, ha az adat része a tipus által jelölt adatnak.

Parameters

adat	- szöveg
tipus	- ahol keressük (vez, ker, cim, mun, pri, bec, beo)

Reimplemented from **Kontakt** (*p.13*).

Definition at line **110** of file **kontakt.cpp**.

```
00110                                                   {
00111     if(tipus == "vez") return get_vez().find(adat) != string::npos;
00112     else if(tipus == "ker") return get_ker().find(adat) != string::npos;
00113     else if(tipus == "cim") return get_cim().find(adat) != string::npos;
00114     else if(tipus == "mun") return get_mun().find(adat) != string::npos;
00115     else if(tipus == "pri") return get_pri().find(adat) != string::npos;
00116     else if(tipus == "bec") return becenev.find(adat) != string::npos;
00117     else {
00118         if(get_vez().find(adat) != string::npos ||
00119            get_ker().find(adat) != string::npos ||
00120            get_cim().find(adat) != string::npos ||
00121            get_mun().find(adat) != string::npos ||
00122            get_pri().find(adat) != string::npos ||
00123            becenev.find(adat) != string::npos) { return true; }
00124     }
00125     return false;
00126 }
```

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

kontakt.hkontakt.cpp

Munkatars Class Reference

Inheritance diagram for Munkatars:

Public Member Functions

Munkatars (string v="", string k="", string c="", string m="", string p="", string b="")
Konstruktor.

string **get_beo** ()
Speciális adat lekérdezése.

void **kiir** (ostream &os=cout)
Munkatars * **clone** ()
Másolat készítése (deep copy).

bool **operator==** (**Kontakt** *m)
bool **string_keres** (string adat, string tipus)

Public Member Functions inherited from Kontakt

Kontakt (string v="", string k="", string c="", string m="", string p="")
Konstruktor.

string **get_vez** ()
Adatok lekérdezése.

string **get_ker** ()
string **get_cim** ()
string **get_mun** ()
string **get_pri** ()

virtual void **kiir** (ostream &os=cout)
virtual **Kontakt** * **clone** ()
Másolat készítése (deep copy).

virtual bool **operator==** (**Kontakt** *k)
virtual **~Kontakt** ()
Destruktor.

virtual bool **string_keres** (string adat, string tipus)

Additional Inherited Members

Protected Attributes inherited from Kontakt

string **vezeteknev**
string **keresztnev**
string **cim**
string **munkahelyi_szam**
string **privat_szam**

Detailed Description

Definition at line **111** of file **kontakt.h**.

Constructor & Destructor Documentation

Munkatars::Munkatars (string *v* = "", string *k* = "", string *c* = "", string *m* = "", string *p* = "", string *b* = "") [inline]

Konstruktor.
Definition at line **115** of file **kontakt.h**.

```
00120                                : Kontakt(v, k, c, p, m), beosztas(b) {}
```

Member Function Documentation

Munkatars * Munkatars::clone () [virtual]

Másolat készítése (deep copy).
Reimplemented from **Kontakt** (*p.12*).
Definition at line **93** of file **kontakt.cpp**.

```
00093 { return new Munkatars(*this); }
```

string Munkatars::get_beo ()

Speciális adat lekérdezése.
Definition at line **19** of file **kontakt.cpp**.

```
00019 { return beosztas; }
```

void Munkatars::kiir (ostream & *os* = cout) [virtual]

Egy munkatárs adatainak kiírása.

Parameters

os	- output stream, default: cout

Reimplemented from **Kontakt** (*p.12*).

Definition at line **42** of file **kontakt.cpp**.

```
00042                                    {
00043     os << "Munkatars;";
00044     (get_vez() != "") ? os << get_vez() << ";" : os << "nincs;";
00045     (get_ker() != "") ? os << get_ker() << ";" : os << "nincs;";
00046     (get_cim() != "") ? os << get_cim() << ";" : os << "nincs;";
00047     (get_mun() != "") ? os << get_mun() << ";" : os << "nincs;";
00048     (get_pri() != "") ? os << get_pri() << ";" : os << "nincs;";
00049     (beosztas != "") ? os << beosztas : os << "nincs";
00050     os << endl;
00051 }
```

bool Munkatars::operator== (Kontakt * *m*) [virtual]

Értékek összehasonlítása.

Parameters

k	- kontakt pointere

Reimplemented from **Kontakt** (*p.13*).

Definition at line **79** of file **kontakt.cpp**.

```
00079                                        {
00080     Munkatars* l = dynamic_cast<Munkatars*> (k);
00081     if(l == NULL) return false;
00082     if(vezeteknev != l->vezeteknev) return false;
00083     if(keresztnev != l->keresztnev) return false;
00084     if(cim != l->cim) return false;
00085     if(munkahelyi_szam != l->munkahelyi_szam) return false;
00086     if(privat_szam != l->privat_szam) return false;
00087     if(beosztas != l->beosztas) return false;
00088     return true;
00089 }
```

bool Munkatars::string_keres (string *adat*, string *tipus*) [virtual]

Szöveg összehasonlító függvény, ami igazat ad vissza, ha az adat része a tipus által jelölt adatnak.

Parameters

adat	- szöveg
tipus	- ahol keressük (vez, ker, cim, mun, pri, bec, beo)

Reimplemented from **Kontakt** (*p.13*).

Definition at line **127** of file **kontakt.cpp**.

```
00127                                                    {
00128     if(tipus == "vez") return get_vez().find(adat) != string::npos;
00129     else if(tipus == "ker") return get_ker().find(adat) != string::npos;
00130     else if(tipus == "cim") return get_cim().find(adat) != string::npos;
00131     else if(tipus == "mun") return get_mun().find(adat) != string::npos;
00132     else if(tipus == "pri") return get_pri().find(adat) != string::npos;
00133     else if(tipus == "beo") return beosztas.find(adat) != string::npos;
00134     else {
00135         if(get_vez().find(adat) != string::npos ||
00136            get_ker().find(adat) != string::npos ||
00137            get_cim().find(adat) != string::npos ||
00138            get_mun().find(adat) != string::npos ||
00139            get_pri().find(adat) != string::npos ||
00140            beosztas.find(adat) != string::npos) { return true; }
00141     }
00142     return false;
00143 }
```

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

kontakt.hkontakt.cpp

5.1.3 Globális függvények

Az egyetlen globális függvény a *kontakt.h* -ban a << operátor, ami a fájlbaírást valósítja meg úgy, hogy meghívja az alosztályhoz tartozó *kiir* függvényt. Paraméterként egy ostream -et és egy *Kontakt* pointert vár.

Function Documentation

ostream & operator<< (ostream & *f*, Kontakt * *k*)

Fájlba írás.

Parameters

f	- output stream
k	- kontakt pointere

Definition at line **53** of file **kontakt.cpp**.

```
00053                                                {
00054     k->kiir(f);
00055     return f;
00056 }
```

5.2 A tároló bemutatása

A lényeges változtatások a tervhez képest ebben az osztályban történtek.

Hogy a feladatleírásnak megfeleljünk, másoló konstruktort és értékadó operátort is kellett készítenünk.

A *hozzaad* és *torol* függvények paraméternek végül referencia helyett pointert kapnak, mert az osztályok == operátorával ellenőrzik, hogy az objektum szerepel-e a nyilvántartásban, ami pedig pointerrel dolgozik. A *torol* akkor ad hibaüzenetet, ha az objektum már eleve nem volt benne a listában, a *hozzaad* pedig akkor, amikor már szerepel a nyilvántartásban egyszer.

A *torol* függvénynek lett egy paraméter nélkül hívható változata is, amely az egész listát kitörli, ezt fogja használni a destruktor is.

Az adatok lekérdezéséhez jött létre a *get_meret*, ami visszaadja a tömb méretét, és a *[]* operátor, ami visszaadja tömb paraméterként kapott indexén lévő *Kontakt* pointert. Indexelési hiba esetén kivételt dob.

A most következtető apró változtatás nem volt szükséges, ellenben egy viszonylag hasznos funkció, ezért nem lett kivéve a programból. A *listaz*

függvény kapott két string paramétert, ezek az *adat* és a *tipus*. Az előbbi helyére beírhatunk valamilyen keresett szöveget, az utóbbinak pedig megadhatjuk hol keressük, vagyis az objektum melyik tulajdonságával hasonlítsa össze (pl. ”vez” → a *vezeteknev* adatban keressük). Ha az *adat* és *tipus* nincs megadva (default esetben ilyenkor mindkettő = ””), akkor mindent kiír, ha csak a *tipus* -t hagyjuk el, akkor az *adat* -ot az egész objektumban keresi, ha csak az *adat* -ot hagyjuk el, akkor csak azokat az adatokat fogja kilistázni, amelyek rendelkeznek *tipus* szerinti tulajdonsággal. Így tehát ez egyfajta beépített szövegkeresőként funkcionál.

A generikus kereséshez végül csak egy sablonfüggvény készült. A korábbiakban említett „leg- elem” kereső azért nem valósult meg, mert a feladat szemponjából nincs értelme ilyet használni. Nem életszerű, hogy valaki azért nyitja ki a telefonkönyvet, hogy megkeresse a leghosszabb névvel rendelkező embert. Sokkal inkább valószínű, hogy fel akarjuk venni a kapcsolatot valakivel, de nem emlékszünk a teljes nevére, csak arra, hogy „A” betűvel kezdődik. Ezenkívül annyi változtatás történt, hogy a *keres* sablonparaméternek csak a *P* predikátumfüggvényt várja és végül nem ad vissza értéket (void típusú), helyette kilistázza a paraméterként kapott ostream -re azokat az objektumokat a tárolóból, amelyekre a predikátumfüggvény true értéket adott vissza.

A *mentes* és a *betoltes* függvények hibaüzenetet adnak, ha a paraméternek kapott fájlt nem lehet megnyitni, de a *betoltes* kivételt is dob, ha hibás bemenet érkezett.

Telefonkonyv Class Reference

Public Member Functions

Telefonkonyv ()
Konstruktor.

Telefonkonyv (**Telefonkonyv** &t)
Másoló konstruktor.

void **hozzaad** (**Kontakt** *k)
void **torol** (**Kontakt** *k)
void **torol** ()
Egész nyilvántartás törlése.

size_t **get_meret** ()
Méret lekérdezése.

void **listaz** (string adat="", string tipus="", ostream &os=cout)
template<typename P > void **keres** (P pre, ostream &os=cout)
void **mentes** (const string &fajl="nyilvantartas.dat")
void **betoltes** (const string &fajl="nyilvantartas.dat")

Telefonkonyv & **operator=** (const **Telefonkonyv** &t)
Kontakt * **operator[]** (size_t i)
~Telefonkonyv ()
Destruktor.

Detailed Description

Definition at line **18** of file **telefonkonyv.h**.

Constructor & Destructor Documentation

Telefonkonyv::Telefonkonyv ()

Konstruktor.

Definition at line **13** of file **telefonkonyv.cpp**.

```
00013                               {
00014     meret = 0;
00015     lista = new Kontakt*[meret];
00016 }
```

Telefonkonyv::Telefonkonyv (Telefonkonyv & *t*)

Másoló konstruktor.

Definition at line **18** of file **telefonkonyv.cpp**.

```
00018                                                {
00019     meret = 0;
00020     lista = new Kontakt*[meret];
00021     *this = t;
00022 }
```

Telefonkonyv::~Telefonkonyv ()

Destruktor.

Definition at line **161** of file **telefonkonyv.cpp**.

```
00161                               {
00162     this->mentes();
00163     this->torol();
00164     delete[] lista;
00165 }
```

Member Function Documentation

void Telefonkonyv::betoltes (const string & *fajl* = `"nyilvantartas.dat"`)

Adatok betöltése fájlból.

Parameters

fajl	- fájl neve, default: "nyilvantartas.dat"

Definition at line **84** of file **telefonkonyv.cpp**.

```
00084                                                  {
00085         fstream f;
00086         f.open(fajl, ios_base::in);
00087         if(!f.is_open()) {
00088             cout << NINCS_FAJL;
00089             return;
00090         }
00091         string input, v, k, c, m, p, b;
```

```
00092         bool hibas_bemenet = false;
00093         while(getline(f, input, ';')) {
00094             if(input == "Kontakt" || input == "Barat" || input == "Munkatars") {
00095                 getline(f, v, ';');
00096                 getline(f, k, ';');
00097                 getline(f, c, ';');
00098                 getline(f, m, ';');
00099                 if(input == "Kontakt") {
00100                     getline(f, p);
00101                     Kontakt* w = new Kontakt(v, k, c, m, p);
00102                     this->hozzaad(w);
00103                     delete w;
00104                 }
00105                 else {
00106                     getline(f, p, ';');
00107                     getline(f, b);
00108                     if(input == "Barat") {
00109                         Barat* w = new Barat(v, k, c, m, p, b);
00110                         this->hozzaad(w);
00111                         delete w;
00112                     }
00113                     else if(input == "Munkatars") {
00114                         Munkatars* w = new Munkatars(v, k, c, m, p, b);
00115                         this->hozzaad(w);
00116                         delete w;
00117                     }
00118                 }
00119             }
00120             else {
00121                 hibas_bemenet = true;
00122                 break;
00123             }
00124         }
00125         f.close();
00126         if(hibas_bemenet) {
00127             this->torol();
00128             stringstream err;
00129             err << "Hibas bemenet erkezett, nem sikerult betolteni az adatokat.." <<
endl;
00130             throw std::out_of_range(err.str());
00131         }
00132 }
```

size_t Telefonkonyv::get_meret ()

Méret lekérdezése.

Definition at line **59** of file **telefonkonyv.cpp**.

```
00059 { return meret; }
```

void Telefonkonyv::hozzaad (Kontakt * *k*)

Elem hozzáadása a nyilvántartáshoz.

Parameters

k	- kontakt pointere

Definition at line **24** of file **telefonkonyv.cpp**.

```
00024                                     {
00025     for(size_t i = 0; i < meret; i++) {
00026         if(*lista[i] == k) {
00027             cout << MAR_VAN;
00028             return;
00029         }
00030     }
00031     Kontakt** uj = new Kontakt*[meret + 1];
00032     for(size_t i = 0; i < meret; i++) {
00033         uj[i] = lista[i];
00034     }
00035     uj[meret++] = k->clone();
00036     delete[] lista;
00037     lista = uj;
00038 }
```

template<typename P > void Telefonkonyv::keres (P *pre*, ostream & *os* = `cout`)`[inline]`

Generikus keresÅ‘fÃ¼ggvÃ©ny.

Parameters

P	- predikátumfüggvény típusa
pre	- predikátumfüggvény
os	- output stream, default: cout

Definition at line **56** of file **telefonkonyv.h**.

```
00056                                           {
00057         size_t cnt = 0;
00058         for(size_t i = 0; i < meret; i++) {
00059             if(pre(lista[i])) {
00060                 lista[i]->kiir(os);
00061                 cnt++;
00062             }
00063         }
00064         if(cnt == 0) cout << "Nincs talalat.." << endl;
00065     }
```

void Telefonkonyv::listaz (string *adat* = `""`, string *tipus* = `""`, ostream & *os* = `cout`)

NyilvÃ¡ntartÃ¡s adatainak listÃ¡zÃ¡sa (kiÃ-ratÃ¡sa), amiben van egy kis keresÅ‘ is, csak azokat az objektumokat listÃ¡zza ki, amiben az "adat" szerepel az objektum "tipus" szerinti (vez, ker, cim, mun, pri, bec, beo) tulajdonsÃ¡gÃ¡ban.

Parameters

adat	- string, default: ""
tipus	- string, default: ""
os	- output stream, default: cout

Definition at line **61** of file **telefonkonyv.cpp**.

```
00061                                                                       {
00062     for(size_t i = 0; i < meret; i++) {
00063         if(lista[i]->string_keres(adat, tipus)) { lista[i]->kiir(os); }
00064     }
00065     if(meret == 0) {
00066         cout << URES_LISTA;
00067         return;
00068     }
00069 }
```

void Telefonkonyv::mentes (const string & *fajl* = `"nyilvantartas.dat"`)

Adatok fájlba írása.

Parameters

fajl	- fájl neve, default: "nyilvantartas.dat"

Definition at line **71** of file **telefonkonyv.cpp**.

```
00071                                               {
00072     fstream f;
00073     f.open(fajl, ios_base::out);
00074     if(!f.is_open()) {
00075         cout << NINCS_FAJL;
00076         return;
00077     }
00078     for(size_t i = 0; i < meret; i++) {
00079         f << lista[i];
00080     }
00081     f.close();
00082 }
```

Telefonkonyv & Telefonkonyv::operator= (const Telefonkonyv & *t*)

Értékadás.

Parameters

t	- telefonkonyv referenciája

Definition at line **134** of file **telefonkonyv.cpp**.

```
00134                                                       {
```

```
00135     if(lista == t.lista) return *this;
00136     delete[] lista;
00137     meret = t.meret;
00138     lista = new Kontakt*[meret];
00139     for(size_t i = 0; i < meret; i++) {
00140         lista[i] = t.lista[i]->clone();
00141     }
00142     return *this;
00143 }
```

Kontakt * Telefonkonyv::operator[] (size_t *i*)

Indexelés.

Parameters

i	- index

Definition at line **145** of file **telefonkonyv.cpp**.

```
00145                                           {
00146     if(i >= meret) {
00147         stringstream err;
00148         err << "Indexelesi hiba.." << endl;
00149         throw std::out_of_range(err.str());
00150     }
00151     return lista[i];
00152 }
```

void Telefonkonyv::torol ()

Egész nyilvántartás törlése.

Definition at line **154** of file **telefonkonyv.cpp**.

```
00154                           {
00155     for(size_t i = 0; i < meret; i++){
00156         delete lista[i];
00157     }
00158     meret = 0;
00159 }
```

void Telefonkonyv::torol (Kontakt * *k*)

Egy elem törlése a nyilvántartásból.

Parameters

k	- kontakt pointere

Definition at line **40** of file **telefonkonyv.cpp**.

```
00040                                       {
00041     int index = -1;
00042     for(size_t i = 0; i < meret; i++) {
00043         if(*lista[i] == k) {
00044             index = i;
00045             break;
00046         }
00047     }
00048     if(index == -1) {
00049         cout << MEG_NINCS;
00050         return;
00051     }
00052     for(size_t i = index; i < meret - 1; i++) {
00053         delete lista[index];
00054         lista[index] = lista[index+1]->clone();
00055     }
00056     delete lista[--meret];
00057 }
```

The documentation for this class was generated from the following files:
telefonkonyv.htelefonkonyv.cpp

5.3 Tesztprogram bemutatása

A telefonkonyv_main.cpp fájlban lévő tesztprogram a standard inputról beolvasott egész szám alapján különböző teszteseteket hajt végre.

A demonstrációs céllal készített tesztprogram nem csak a standard inputról olvas, hanem egy .dat fájlból is.

A kivételeket a főosztály kezeli. A saját kivételosztály teszteléséhez létrehozott osztály a következő:

Hiba Class Reference

Kivételosztály a tesztesetekhez.

Public Member Functions

Hiba (const string &)

Detailed Description

Kivételosztály a tesztesetekhez.

Definition at line **16** of file **telefonkonyv_main.cpp**.

Constructor & Destructor Documentation

Hiba::Hiba (const string &) [inline]

Parameters

-	nem használjuk semmire

Definition at line **19** of file **telefonkonyv_main.cpp**.

```
00016 class Hiba {
00017 public:
00019     Hiba(const string&) {}
00020 };
```

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

telefonkonyv_main.cpp

5.3.1 Az egyes teszteseteket megvalósító függvények:

void teszt_1 ()

TESZT 1 fájlból beolvassa a tároló adatait, majd azokat kiírja a standard outputra.

void teszt_2 ()

TESZT 2 hozzáad a listához néhány kontaktot, ezután kiírja ezeket a standard outputra, majd töröl pár adatot és ismét kiír, végül az egész listát törli és ismét megpróbál kiírni.

void teszt_3 ()

TESZT 3 fájlból beolvas egy tárolóba, majd másoló konstuktorral és értékadó operátorral csinál egy-egy új tárolót és mindhárom elemeit kiírja.

void teszt_4 ()

TESZT 4 fájlból beolvas egy tárolóba, kiírja az adatokat, majd keresi a "B" betűvel kezdődő vezetéknevűeket, kiírja a találatokat aztán azokat a munkatársakat keresi és listázza ki, akik postásként dolgoznak.

6. Tesztelés

A feladat során elkészített dinamikus nyilvántartás tesztjeit a következő szempontok szerint terveztem meg:

- Interfész teszt
  - Paraméterek tesztje
  - Generikus működés tesztje
- Funkcionális teszt
  - Tudjon fájlba menteni és fájlból betölteni
  - Tudjon adatokat felvenni, törölni, listázni
  - Tudjon tárolókat másolni, értékül adni
  - Használja a generikus keresést valamilyen formában
  - A tesztesetek legyenek olyanok, hogy minél több hiba is előforduljon, ezzel a program hibajelzésit ellenőrizve

6.1 Interfész teszt

A tesztprogram több különböző adatra, különböző paraméterekkel vizsgálja az elkészített osztályokat. A fordítás hibamentes, ami az interfész tesztnek való megfelelést igazolja.

6.2 A funkcionális tesztek

teszt_1

```
TESZT 1.

---[Beolvasas egy nem letezo fajlbol]---

A fajl nem letezik vagy nem sikerult megnyitni..

---[Beolvasas egy valid fajlbol es listazas]---

Ez a kontakt mar szerepel a listaban..
```

Kontakt;Boro;Zoltan;Budapest Szent Gellert ter 91.;nincs;+36 31 561 1659
Barat;Bac;Ilus;Miske Dayka Gabor utca 46.;+36 20 792 9904;+36 70 803 4653;Baci
Munkatars;Bubo;Bubo;Vese Nanasi ut 62.;+36 31 274 8403;+36 50 193 9592;Haziorvos
Munkatars;Lev;Elek;Atany Erzsebet ter 90.;+36 31 292 6696;+36 20 861 0424;Postas
Barat;Bac;Ilus;Miske Dayka Gabor utca 46.;+36 70 803 4653;+36 20 792 9904;Baci
Munkatars;Bubo;Bubo;Vese Nanasi ut 62.;+36 50 193 9592;+36 31 274 8403;Haziorvos
Munkatars;Lev;Elek;Atany Erzsebet ter 90.;+36 20 861 0424;+36 31 292 6696;Postas

teszt_2

TESZT 2.

---[Uj kontaktok felvetele a listara majd kiiras]---

Ez a kontakt mar szerepel a listaban..
Kontakt;Boro;Zoltan;Budapest Szent Gellert ter 91.;nincs;+36 31 561 1659
Barat;Bac;Ilus;Miske Dayka Gabor utca 46.;+36 70 803 4653;+36 20 792 9904;Baci
Munkatars;Bubo;Bubo;Vese Nanasi ut 62.;+36 50 193 9592;+36 31 274 8403;Haziorvos
Munkatars;Lev;Elek;Atany Erzsebet ter 90.;+36 20 861 0424;+36 31 292 6696;Postas

---[Kontaktok torlese listarol majd kiiras]---

Ez a kontakt nincs is a listaban..
Kontakt;Boro;Zoltan;Budapest Szent Gellert ter 91.;nincs;+36 31 561 1659
Munkatars;Lev;Elek;Atany Erzsebet ter 90.;+36 20 861 0424;+36 31 292 6696;Postas

---[Lista torlese majd kiiras]---

A nyilvantartasban nincs adat..

teszt_3

TESZT 3.

---[Elso tarolo betoltese fajlbol]---
Kontakt;Boro;Zoltan;Budapest Szent Gellert ter 91.;nincs;+36 31 561 1659
Barat;Bac;Ilus;Miske Dayka Gabor utca 46.;+36 20 792 9904;+36 70 803 4653;Baci
Munkatars;Bubo;Bubo;Vese Nanasi ut 62.;+36 31 274 8403;+36 50 193 9592;Haziorvos
Munkatars;Lev;Elek;Atany Erzsebet ter 90.;+36 31 292 6696;+36 20 861 0424;Postas

---[Masodik tarolo masolo konstruktorral]---
Kontakt;Boro;Zoltan;Budapest Szent Gellert ter 91.;nincs;+36 31 561 1659

```
Barat;Bac;Ilus;Miske Dayka Gabor utca 46.;+36 20 792 9904;+36 70 803 4653;Baci
Munkatars;Bubo;Bubo;Vese Nanasi ut 62.;+36 31 274 8403;+36 50 193 9592;Haziorvos
Munkatars;Lev;Elek;Atany Erzsebet ter 90.;+36 31 292 6696;+36 20 861 0424;Postas

---[Harmadik tarolo ertekadassal]---
Kontakt;Boro;Zoltan;Budapest Szent Gellert ter 91.;nincs;+36 31 561 1659
Barat;Bac;Ilus;Miske Dayka Gabor utca 46.;+36 20 792 9904;+36 70 803 4653;Baci
Munkatars;Bubo;Bubo;Vese Nanasi ut 62.;+36 31 274 8403;+36 50 193 9592;Haziorvos
Munkatars;Lev;Elek;Atany Erzsebet ter 90.;+36 31 292 6696;+36 20 861 0424;Postas
```

teszt_4

```
TESZT 4.

---[B betuvel kezdodo vezeteknevuek]---
Kontakt;Boro;Zoltan;Budapest Szent Gellert ter 91.;nincs;+36 31 561 1659
Barat;Bac;Ilus;Miske Dayka Gabor utca 46.;+36 20 792 9904;+36 70 803 4653;Baci
Munkatars;Bubo;Bubo;Vese Nanasi ut 62.;+36 31 274 8403;+36 50 193 9592;Haziorvos
Munkatars;Lev;Elek;Atany Erzsebet ter 90.;+36 31 292 6696;+36 20 861 0424;Postas

---[B betuvel kezdodo vezeteknevuek]---
Kontakt;Boro;Zoltan;Budapest Szent Gellert ter 91.;nincs;+36 31 561 1659
Barat;Bac;Ilus;Miske Dayka Gabor utca 46.;+36 20 792 9904;+36 70 803 4653;Baci
Munkatars;Bubo;Bubo;Vese Nanasi ut 62.;+36 31 274 8403;+36 50 193 9592;Haziorvos

---[Postas hivatasuak]---
Munkatars;Lev;Elek;Atany Erzsebet ter 90.;+36 31 292 6696;+36 20 861 0424;Postas

0
```

6.3 Memóriakezelés tesztje

A memóriakezelés ellenőrzését a laborgyakorlaton is használt MEMTRACE modullal végeztem. Ehhez include -oltam a ”memtrace.h” állományt a standard fejlécállományok után. Memóriakezelési hibát nem tapasztaltam a futtatások során.

6.4 Lefedettségi teszt

A Jporta rendszer néhány kódrészletet megjelölt, amelyek nem futottak a programban. Ezek a következők:

- A *get_meret* és az *operator[]* végül nem lettek használva

- A *mentes* függvény utolsó két sora, ami akkor fut le, ha a fájlt nem lehet megnyitni
- A *betoltes* függvény utolsó pár sora, ami a hibás bemenetre dobna kivételt
- A predikátumfüggvények közül egy pár, ami nem lett használva, mert a ezek közül csak kettőt használtunk
- A *get_bec* függvény, ami a becenev adatot adja vissza
- Valamint a *string_keres* függvény pár sora, mert azt csak default esetben alkalmaztuk

get_meret

```
59. size_t Telefonkonyv::get_meret() { return meret; }
```

operator[]

```
145. Kontakt* Telefonkonyv::operator[](size_t i) {
146.     if(i >= meret) {
147.         stringstream err;
148.         err << "Indexelesi hiba.." << endl;
149.         throw std::out_of_range(err.str());
150.     }
151.     return lista[i];
```

mentes

```
75.         cout << NINCS_FAJL;
76.         return;
77.     }
```

betoltes

```
121.                 hibas_bemenet = true;
122.                 break;
123.             }
124.         }
125.         f.close();
126.         if(hibas_bemenet) {
127.             this->torol();
128.             stringstream err;
129.             err << "Hibas bemenet erkezett, nem sikerult betolteni az adatokat.." << endl;
130.             throw std::out_of_range(err.str());
```

predikátumfüggvények

```
167. bool A_kezd_vez(Kontakt* k) { return (k->get_vez()[0] == 'A'); }
168. bool B_kezd_vez(Kontakt* k) { return (k->get_vez()[0] == 'B'); }
169. bool K_kezd_vez(Kontakt* k) { return (k->get_vez()[0] == 'K'); }
170. bool N_kezd_vez(Kontakt* k) { return (k->get_vez()[0] == 'N'); }
171. bool tanar(Kontakt* k) {
172.     Munkatars* m = dynamic_cast<Munkatars*>(k);
173.     return (m && m->get_beo() == "Tanar");
```

get_bec

```
18.  string Barat::get_bec() { return becenev; }
```

string_keres

```
103.         keresztnev.find(adat) != string::npos ||
104.         cim.find(adat) != string::npos ||
105.         munkahelyi_szam.find(adat) != string::npos ||
106.         privat_szam.find(adat) != string::npos) { return true; }
107.     }
108.     return false;
```

```
125.     return false;
```

```
142.     return false;
```

7. Mellékletek

7.1 kontakt.cpp

```
00011 #include "kontakt.h"
00012
00013 string Kontakt::get_vez() { return vezeteknev; }
00014 string Kontakt::get_ker() { return keresztnev; }
00015 string Kontakt::get_cim() { return cim; }
00016 string Kontakt::get_mun() { return munkahelyi_szam; }
00017 string Kontakt::get_pri() { return privat_szam; }
00018 string Barat::get_bec() { return becenev; }
00019 string Munkatars::get_beo() { return beosztas; }
00020
00021 void Kontakt::kiir(ostream& os) {
00022     os << "Kontakt;";
00023     (vezeteknev != "") ? os << vezeteknev << ";" : os << "nincs;";
00024     (keresztnev != "") ? os << keresztnev << ";" : os << "nincs;";
00025     (cim != "") ? os << cim << ";" : os << "nincs;";
00026     (munkahelyi_szam != "") ? os << munkahelyi_szam << ";" : os << "nincs;";
00027     (privat_szam != "") ? os << privat_szam : os << "nincs";
00028     os << endl;
00029 }
00030
00031 void Barat::kiir(ostream& os) {
00032     os << "Barat;";
00033     (get_vez() != "") ? os << get_vez() << ";" : os << "nincs;";
00034     (get_ker() != "") ? os << get_ker() << ";" : os << "nincs;";
00035     (get_cim() != "") ? os << get_cim() << ";" : os << "nincs;";
00036     (get_mun() != "") ? os << get_mun() << ";" : os << "nincs;";
00037     (get_pri() != "") ? os << get_pri() << ";" : os << "nincs;";
00038     (becenev != "") ? os << becenev : os << "nincs";
00039     os << endl;
00040 }
00041
00042 void Munkatars::kiir(ostream& os) {
00043     os << "Munkatars;";
00044     (get_vez() != "") ? os << get_vez() << ";" : os << "nincs;";
00045     (get_ker() != "") ? os << get_ker() << ";" : os << "nincs;";
00046     (get_cim() != "") ? os << get_cim() << ";" : os << "nincs;";
00047     (get_mun() != "") ? os << get_mun() << ";" : os << "nincs;";
00048     (get_pri() != "") ? os << get_pri() << ";" : os << "nincs;";
00049     (beosztas != "") ? os << beosztas : os << "nincs";
00050     os << endl;
00051 }
00052
00053 ostream& operator<<(ostream& f, Kontakt* k) {
00054     k->kiir(f);
00055     return f;
00056 }
```

```
00057
00058 bool Kontakt::operator==(Kontakt* k) {
00059     if(vezeteknev != k->vezeteknev) return false;
00060     if(keresztnev != k->keresztnev) return false;
00061     if(cim != k->cim) return false;
00062     if(munkahelyi_szam != k->munkahelyi_szam) return false;
00063     if(privat_szam != k->privat_szam) return false;
00064     return true;
00065 }
00066
00067 bool Barat::operator==(Kontakt* k) {
00068     Barat* l = dynamic_cast<Barat*> (k);
00069     if(l == NULL) return false;
00070     if(vezeteknev != l->vezeteknev) return false;
00071     if(keresztnev != l->keresztnev) return false;
00072     if(cim != l->cim) return false;
00073     if(munkahelyi_szam != l->munkahelyi_szam) return false;
00074     if(privat_szam != l->privat_szam) return false;
00075     if(becenev != l->becenev) return false;
00076     return true;
00077 }
00078
00079 bool Munkatars::operator==(Kontakt* k) {
00080     Munkatars* l = dynamic_cast<Munkatars*> (k);
00081     if(l == NULL) return false;
00082     if(vezeteknev != l->vezeteknev) return false;
00083     if(keresztnev != l->keresztnev) return false;
00084     if(cim != l->cim) return false;
00085     if(munkahelyi_szam != l->munkahelyi_szam) return false;
00086     if(privat_szam != l->privat_szam) return false;
00087     if(beosztas != l->beosztas) return false;
00088     return true;
00089 }
00090
00091 Kontakt* Kontakt::clone() { return new Kontakt(*this); }
00092 Barat* Barat::clone() { return new Barat(*this); }
00093 Munkatars* Munkatars::clone() { return new Munkatars(*this); }
00094
00095 bool Kontakt::string_keres(string adat, string tipus) {
00096     if(tipus == "vez") return vezeteknev.find(adat) != string::npos;
00097     else if(tipus == "ker") return keresztnev.find(adat) != string::npos;
00098     else if(tipus == "cim") return cim.find(adat) != string::npos;
00099     else if(tipus == "mun") return munkahelyi_szam.find(adat) != string::npos;
00100     else if(tipus == "pri") return privat_szam.find(adat) != string::npos;
00101     else {
00102         if(vezeteknev.find(adat) != string::npos ||
00103            keresztnev.find(adat) != string::npos ||
00104            cim.find(adat) != string::npos ||
00105            munkahelyi_szam.find(adat) != string::npos ||
00106            privat_szam.find(adat) != string::npos) { return true; }
00107     }
00108     return false;
00109 }
00110 bool Barat::string keres(string adat, string tipus) {
00111     if(tipus == "vez") return get_vez().find(adat) != string::npos;
00112     else if(tipus == "ker") return get_ker().find(adat) != string::npos;
00113     else if(tipus == "cim") return get_cim().find(adat) != string::npos;
00114     else if(tipus == "mun") return get_mun().find(adat) != string::npos;
00115     else if(tipus == "pri") return get_pri().find(adat) != string::npos;
00116     else if(tipus == "bec") return becenev.find(adat) != string::npos;
00117     else {
00118         if(get_vez().find(adat) != string::npos ||
00119            get_ker().find(adat) != string::npos ||
00120            get_cim().find(adat) != string::npos ||
00121            get_mun().find(adat) != string::npos ||
00122            get pri().find(adat) != string::npos ||
00123            becenev.find(adat) != string::npos) { return true; }
00124     }
00125     return false;
00126 }
00127 bool Munkatars::string_keres(string adat, string tipus) {
00128     if(tipus == "vez") return get_vez().find(adat) != string::npos;
00129     else if(tipus == "ker") return get_ker().find(adat) != string::npos;
00130     else if(tipus == "cim") return get_cim().find(adat) != string::npos;
00131     else if(tipus == "mun") return get_mun().find(adat) != string::npos;
00132     else if(tipus == "pri") return get_pri().find(adat) != string::npos;
00133     else if(tipus == "beo") return beosztas.find(adat) != string::npos;
```

```
00134     else {
00135         if(get_vez().find(adat) != string::npos ||
00136            get_ker().find(adat) != string::npos ||
00137            get_cim().find(adat) != string::npos ||
00138            get_mun().find(adat) != string::npos ||
00139            get_pri().find(adat) != string::npos ||
00140            beosztas.find(adat) != string::npos) { return true; }
00141     }
00142     return false;
00143 }
```

7.2 telefonkonyv.cpp

```
00011 #include "telefonkonyv.h"
00012
00013 Telefonkonyv::Telefonkonyv() {
00014     meret = 0;
00015     lista = new Kontakt*[meret];
00016 }
00017
00018 Telefonkonyv::Telefonkonyv(Telefonkonyv& t) {
00019     meret = 0;
00020     lista = new Kontakt*[meret];
00021     *this = t;
00022 }
00023
00024 void Telefonkonyv::hozzaad(Kontakt* k) {
00025     for(size_t i = 0; i < meret; i++) {
00026         if(*lista[i] == k) {
00027             cout << MAR_VAN;
00028             return;
00029         }
00030     }
00031     Kontakt** uj = new Kontakt*[meret + 1];
00032     for(size_t i = 0; i < meret; i++) {
00033         uj[i] = lista[i];
00034     }
00035     uj[meret++] = k->clone();
00036     delete[] lista;
00037     lista = uj;
00038 }
00039
00040 void Telefonkonyv::torol(Kontakt* k) {
00041     int index = -1;
00042     for(size_t i = 0; i < meret; i++) {
00043         if(*lista[i] == k) {
00044             index = i;
00045             break;
00046         }
00047     }
00048     if(index == -1) {
00049         cout << MEG_NINCS;
00050         return;
00051     }
00052     for(size_t i = index; i < meret - 1; i++) {
00053         delete lista[index];
00054         lista[index] = lista[index+1]->clone();
00055     }
00056     delete lista[--meret];
00057 }
00058
00059 size_t Telefonkonyv::get_meret() { return meret; }
00060
00061 void Telefonkonyv::listaz(string adat, string tipus, ostream& os) {
00062     for(size_t i = 0; i < meret; i++) {
00063         if(lista[i]->string_keres(adat, tipus)) { lista[i]->kiir(os); }
00064     }
00065     if(meret == 0) {
00066         cout << URES_LISTA;
00067         return;
00068     }
```

```
00069 }
00070
00071 void Telefonkonyv::mentes(const string& fajl) {
00072     fstream f;
00073     f.open(fajl, ios_base::out);
00074     if(!f.is_open()) {
00075         cout << NINCS_FAJL;
00076         return;
00077     }
00078     for(size_t i = 0; i < meret; i++) {
00079         f << lista[i];
00080     }
00081     f.close();
00082 }
00083
00084 void Telefonkonyv::betoltes(const string& fajl) {
00085         fstream f;
00086         f.open(fajl, ios_base::in);
00087         if(!f.is_open()) {
00088             cout << NINCS_FAJL;
00089             return;
00090         }
00091         string input, v, k, c, m, p, b;
00092         bool hibas_bemenet = false;
00093         while(getline(f, input, ';')) {
00094             if(input == "Kontakt" || input == "Barat" || input == "Munkatars") {
00095                 getline(f, v, ';');
00096                 getline(f, k, ';');
00097                 getline(f, c, ';');
00098                 getline(f, m, ';');
00099                 if(input == "Kontakt") {
00100                     getline(f, p);
00101                     Kontakt* w = new Kontakt(v, k, c, m, p);
00102                     this->hozzaad(w);
00103                     delete w;
00104                 }
00105                 else {
00106                     getline(f, p, ';');
00107                     getline(f, b);
00108                     if(input == "Barat") {
00109                         Barat* w = new Barat(v, k, c, m, p, b);
00110                         this->hozzaad(w);
00111                         delete w;
00112                     }
00113                     else if(input == "Munkatars") {
00114                         Munkatars* w = new Munkatars(v, k, c, m, p, b);
00115                         this->hozzaad(w);
00116                         delete w;
00117                     }
00118                 }
00119             }
00120             else {
00121                 hibas_bemenet = true;
00122                 break;
00123             }
00124         }
00125         f.close();
00126         if(hibas_bemenet) {
00127             this->torol();
00128             stringstream err;
00129             err << "Hibas bemenet erkezett, nem sikerult betolteni az adatokat.." <<
endl;
00130             throw std::out_of_range(err.str());
00131         }
00132 }
00133
00134 Telefonkonyv& Telefonkonyv::operator=(const Telefonkonyv& t) {
00135     if(lista == t.lista) return *this;
00136     delete[] lista;
00137     meret = t.meret;
00138     lista = new Kontakt*[meret];
00139     for(size_t i = 0; i < meret; i++) {
00140         lista[i] = t.lista[i]->clone();
00141     }
00142     return *this;
00143 }
00144
```

```
00145 Kontakt* Telefonkonyv::operator[](size_t i) {
00146     if(i >= meret) {
00147         stringstream err;
00148         err << "Indexelesi hiba.." << endl;
00149         throw std::out_of_range(err.str());
00150     }
00151     return lista[i];
00152 }
00153
00154 void Telefonkonyv::torol() {
00155     for(size_t i = 0; i < meret; i++){
00156         delete lista[i];
00157     }
00158     meret = 0;
00159 }
00160
00161 Telefonkonyv::~Telefonkonyv() {
00162     this->mentes();
00163     this->torol();
00164     delete[] lista;
00165 }
00166
00167 bool A_kezd_vez(Kontakt* k) { return (k->get_vez()[0] == 'A'); }
00168 bool B_kezd_vez(Kontakt* k) { return (k->get_vez()[0] == 'B'); }
00169 bool K_kezd_vez(Kontakt* k) { return (k->get_vez()[0] == 'K'); }
00170 bool N_kezd_vez(Kontakt* k) { return (k->get_vez()[0] == 'N'); }
00171 bool tanar(Kontakt* k) {
00172     Munkatars* m = dynamic_cast<Munkatars*>(k);
00173     return (m && m->get_beo() == "Tanar");
00174 }
00175 bool postas(Kontakt* k) {
00176     Munkatars* m = dynamic_cast<Munkatars*>(k);
00177     return (m && m->get_beo() == "Postas");
00178 }
```

7.3 telefonkonyv_main.cpp

```
00011 #include <iostream>
00012 #include "memtrace.h"
00013 #include "telefonkonyv.h"
00014
00016 class Hiba {
00017 public:
00019     Hiba(const string&) {}
00020 };
00021
00025 void teszt_1() {
00026     Telefonkonyv t;
00027     t.betoltes("teszt.dat");
00028     t.mentes("real.dat");
00029     // Itt megtörtént egy mentés is
00030
00031     Telefonkonyv t2;
00032
00033     cout << TESZT_1_FAKE;
00034     t.betoltes("fake.dat");
00035     // Itt egy nem lÃ©tezÅ‘ fÃ¡jlbÃ³l prÃ³bÃ¡lunk betÃ¶lteni adatokat, hibaÃ¼zenetet
fogunk rÃ¡ kapni
00036
00037     cout << TESZT_1_REAL;
00038     t.betoltes("real.dat");
00039     t.listaz();
00040 }
00041
00047 void teszt_2() {
00048     Telefonkonyv t;
00049     Kontakt* k = new Kontakt("Boro", "Zoltan", "Budapest Szent Gellert ter 91.", "",
"+36 31 561 1659");
00050     Barat* b = new Barat("Bac", "Ilus", "Miske Dayka Gabor utca 46.", "+36 20 792 9904",
"+36 70 803 4653", "Baci");
00051     Munkatars* m = new Munkatars("Bubo", "Bubo", "Vese Nanasi ut 62.", "+36 31 274
8403", "+36 50 193 9592", "Haziorvos");
00052     Munkatars* m2 = new Munkatars("Lev", "Elek", "Atany Erzsebet ter 90.", "+36 31 292
6696", "+36 20 861 0424", "Postas");
```

```
00053
00054     cout << TESZT_2_ADD;
00055     t.hozzaad(k);
00056     // Zoltán jelenleg munkát keres, ezért jelenleg nincsen munkahelyi száma,
00057     // ezt listázásnál látni is fogjuk
00058     t.hozzaad(b);
00059     t.hozzaad(m);
00060     t.hozzaad(m2);
00061     t.hozzaad(m2);
00062     // Eleket szándékosan kétszer próbáljuk meg felvenni a nyilvántartásba,
00063     // azonban amint majd látjuk is csak egyszer fog szerepelni a listában
00064     t.listaz();
00065     cout << endl;
00066
00067     cout << TESZT_2_REMOVE;
00068     t.torol(m);
00069     t.torol(b);
00070     t.torol(b);
00071     // Ilust véletlenül ismét törölni próbáljuk, pedig ekkor már nem szerepel a listán,
itt is hibaüzenetet kapunk
00072     t.listaz();
00073     cout << endl;
00074
00075     cout << TESZT_2_DELETE;
00076     t.torol();
00077     // A nyilvántartás most teljesen üres, itt is hibaüzenetet fogunk kapni
00078     t.listaz();
00079     cout << endl;
00080
00081     delete k;
00082     delete b;
00083     delete m;
00084     delete m2;
00085 }
00086
00091 void teszt_3() {
00092     cout << TESZT_3_LOAD;
00093     Telefonkonyv t;
00094     t.betoltes("teszt.dat");
00095     t.listaz();
00096
00097     cout << TESZT_3_COPY;
00098     Telefonkonyv t2 = t;
00099     t2.listaz();
00100
00101     cout << TESZT_3_VAL;
00102     Telefonkonyv t3;
00103     t3 = t;
00104     t3.listaz();
00105 }
00106
00107
00112 void teszt_4() {
00113     Telefonkonyv t;
00114     t.betoltes("teszt.dat");
00115
00116     cout << TESZT_4_MIND;
00117     t.listaz();
00118
00119     cout << TESZT 4 B;
00120     t.keres(B_kezd_vez);
00121
00122     cout << TESZT_4_POSTAS;
00123     t.keres(postas);
00124
00125 }
00126
00129 int main() {
00130     cout << START;
00131     int teszteset;
00132     while (cin >> teszteset && teszteset > 0) {
00133         try {
00134             switch (teszteset) {
00135                 case 1:
00136                     cout << "TESZT 1." << endl; // mentés, betöltés és kiírás tesztelése
00137                     teszt_1();
00138                     cout << endl;
```

```
00139                     break;
00140                 case 2:
00141                     cout << "TESZT 2." << endl; // hozzáadás és törlés tesztelése
00142                     teszt_2();
00143                     cout << endl;
00144                     break;
00145                 case 3:
00146                     cout << "TESZT 3." << endl; // másolás és értékadás tesztelése
00147                     teszt_3();
00148                     cout << endl;
00149                     break;
00150                 case 4:
00151                     cout << "TESZT 4." << endl; // generikus keresés tesztelése
00152                     teszt_4();
00153                     cout << endl;
00154                     break;
00155                 default:
00156                     cout << NINCS_TESZT;
00157             }
00158         // kivételkezelés
00159         } catch (std::exception& e) {
00160             cerr << e.what() << endl;
00161         } catch (Hiba&) {
00162             cerr << SAJAT_KIVETEL << endl;
00163         } catch (...) {
00164             cerr << NAGY_BAJ << endl;
00165         }
00166     }
00167 
00168     return 0;
00169 }
```