# **Nagy házi feladat**

A programozás alapjai 2.

2019-2020/2

**Telefonkönyv program**

Végleges dokumentáció

**Tóth Ádám László**

**TK6NT3**

2020.05.16

**Tartalomjegyzék**

[**Nagy házi feladat** 1](#_Toc40538321)

[**1. Feladatkiírás** 3](#_Toc40538322)

[**2. Feladatspecifikáció** 3](#_Toc40538323)

[**3. Feladathoz készített terv** 4](#_Toc40538324)

[**4. Objektummodell** 5](#_Toc40538325)

[**4.1 Objektumok ismertetése** 6](#_Toc40538326)

[**4.2 Algoritmusok** 9](#_Toc40538327)

[**5. Megvalósítás** 9](#_Toc40538328)

[**5.1 Tesztesetek leírása** 10](#_Toc40538329)

[**6. Tesztelés** 12](#_Toc40538330)

[**6.1 A tesztesetek eredményei** 12](#_Toc40538331)

[**6.2 A memóriakezelés ellenőrzésének eredménye** 16](#_Toc40538332)

[**6.3 Lefedettségi teszt** 16](#_Toc40538333)

[**7. Mellékletek** 16](#_Toc40538334)

[**7.1 main.cpp** 17](#_Toc40538335)

[**7.2 phonebook.cpp** 20](#_Toc40538336)

[**7.3 phonebook.hpp** 24](#_Toc40538337)

[**7.4 contact.hpp** 25](#_Toc40538338)

[**7.5 array.hpp** 26](#_Toc40538339)

[**7.6 stored\_types.hpp** 28](#_Toc40538340)

[**7.7 string.cpp** 30](#_Toc40538341)

[**7.8 string.h** 32](#_Toc40538342)

## **1. Feladatkiírás**

**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 **Telefonkönyv** program egy egyszerű, adatbáziskezelő jellegű program, objektum orientált alapokon megvalósítva.   
Többek között képes a fent említett adatok felvételére szabványos bemenetről, azok listázására és adott adatok törlésére. Ezen felül a felhasználó kereshet az eltárolt adatok között, és módosíthatja az általa választott adatokat.   
A program kezelése karakteres felületen keresztül történik. Az adatok funkciójuknak megfelelően szöveges formában, dinamikusan kezelt memóriaterületen kerülnek eltárolásra. A keresés során a felhasználó az általa választott paraméterek (eltárolt adatok) között kedvére kereshet.  
A feladat részeként elkészítek egy tesztprogramot is, ami szemlélteti a megvalósított program funkcionalitását.

## **3. Feladathoz készített terv**

**Telefonkönyv program**

A mellékelt ábrán látható a programhoz tervezett osztályok kapcsolata. Az **Array** nevű osztály hasonló az előadáson is előkerült generikus tömbhöz, miközben a **String** osztály úgyszintén ismerős lehet a korábbi foglalkozásokról. A **String** osztály feladata magától értetődő, karakterláncokat tárol és kezel, a megszokott módon. Az **Array** osztály feladata adott típusú elemek eltárolása. A “tömb” sablonnal kerül megvalósításra, így valamennyi adattípussal, többek között ebben az esetben a **Contact** osztály példányaival is kompatibilis.

A **Contact** osztály rendeli egymáshoz az egy bejegyzésben tárolt adatokat, amik a **Name**, az **Address**, a **PhNum** vagy a **Date** osztályok által kerülnek eltárolásra. Ezeknek a kis osztályoknak a célja az adatok nyilvántartási szempontból való elkülönítése és kezelése.

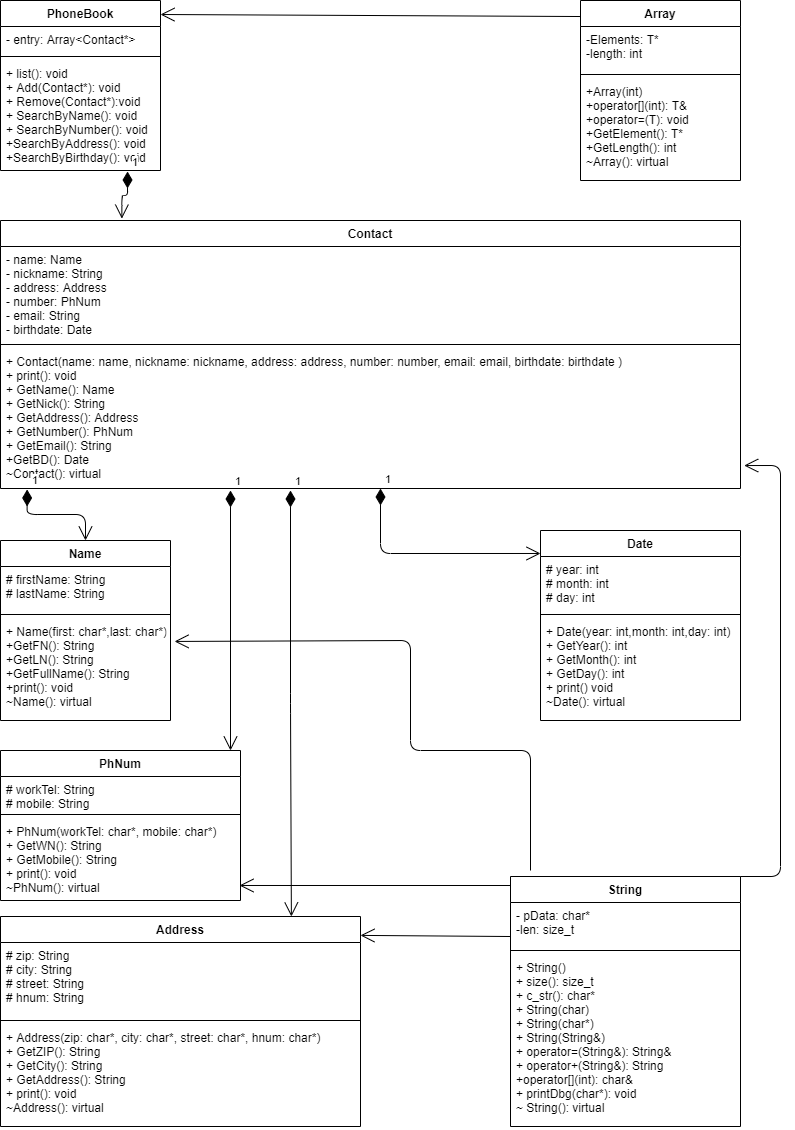
A **PhoneBook** osztály tartalmazza a **Contact** osztály példányaiból felépülő **Array** típusú tömböt.

Tehát a terv szerint egy bejegyzésben eltárolt adatok a következőek:

**Name** (Firstname,Lastname), **Nickname**, **Address** (ZIP, City, Street, Housenumber), **Number** (Work number, Mobile number), **Email address**, **Date of Birth** (Year, Month, Day)

A dátumot leszámítva minden adat dinamikusan, szövegként kerül eltárolásra.

## **4. Objektummodell**

****

### **4.1 Objektumok ismertetése**

#### **4.1.1 Address – Lakcím tárolása**

Az Address osztály négy String típusú adattagot foglal magába. Tagfüggvényei egyszerű adatlekérdező függvények. Továbbá tartalmaz egy print() tagfüggvényt, ami a tesztelés során a standard outputra írja a lakcímet, formázva.

#### **4.1.2 Array – Generikus tömb**

Az Array osztály egy sablonnal megvalósított generikus tömb. Az Elements adattag tárolja az elemek pointereit, a length tag pedig az elemszám elkönyvelésére szolgál. A szokásos kezelést segítő függvényeken kívül rendelkezik egy bovit() függvénnyel, es egy elem\_ki() függvénnyel. Ezek a paraméterül kapott elem pointert adják hozzá/törlik a tömb egy adott példányából.

#### **4.1.3 Contact – Az összetartozó adatok egymáshoz rendelése**

Ebben az osztályban tároljuk az egy névjegyhez tartozó adatokat, és ilyen elemekből épül fel a tömb, ami a telefonkönyvet alkotja. Tartalmaz egy név, becenév, lakcím, telefonszám, email cím és születési dátum adattagot. Ezek közül mindegyikhez tartozik egy Get-Set függvénypár az adatok kezeléséhez.

#### **4.1.4 Date – Dátum eltárolása**

Egyszerű osztály 3 egész szám eltárolására. A tagfüggvényei megegyeznek az Address osztály tagfüggvényeivel.

#### **4.1.5 Name – A névjegyhez tartozó név**

Egyszerű osztály 2 String típusú adat eltárolására. A tagfüggvényei megegyeznek az Address osztály tagfüggvényeivel.

#### **4.1.6 PhNum - Telefonszámok**

Egyszerű osztály 2 String típusú adat eltárolására. A tagfüggvényei megegyeznek az Address osztály tagfüggvényeivel.

#### **4.1.7 PhoneBook – Maga a telefonkönyv**

Ez az osztály tartalmazza azt Array elemet, ami Contact típusú elemeket tárol, a telefonkönyvvel való interakció ezen az osztályon keresztül történik. Az Add() függvény hozzáadja a paraméterül kapott Contact pointert a tömbhöz. A Remove() eltávolítja ugyanezt a tömbből. Az osztálynak van egy list\_contacts() tagfüggvénye, ami meg hívja a Contact osztály elemeinek print() tagfüggvényeit, megjelenítve azokat a kimeneten. A SearchBy…() függvények a paraméterül kapott elemet keresik a tömbben, találat esetén kilistázzák az adatait. A Change…() függvények egy adott nevű bejegyzés megfelelő paraméterét cserélik le arra amit kaptak. Az Inp…() függvények a nevüknek megfelelő típusú adatot hoznak létre a szabványos bemeneten keresztül kapott adatról.

#### **4.1.8 String – Generikus szöveges adattípus**

Ez az osztály megegyezik a laborgyakorlatokon elkészített String osztállyal, ezért a tagjait most nem részletezném.

### **4.2 Algoritmusok**

A program nem tartalmaz összetett algoritmust. A keresőfüggvények a tömb bejárása során pontos egyezést keresnek a kapott paraméterrel.

## **5. Megvalósítás**

A program megírásával nem tértem el az eredeti specifikációban leírtaktól. A korábban elkészített objektummodellhez képest viszont bekövetkeztek apróbb változtatások, valamint szükség merült fel újabb tagfüggvények implementálására, ezeket feljebb ismertettem. Az elkészített tesztprogram a main.cpp állományban található, nem használtam fel hozzá további függvényeket, a kitalált tesztesetek kódjai sorban követik egymást. A program elkészítése során a forráskódot nem láttam el megjegyzésekkel. A konzolos felületen megjelenő kimenetet próbáltam esztétikusra és átláthatóra alakítani, több-kevesebb sikerrel.

### **5.1 Tesztesetek leírása**

**T1** – A program kódjában felvettem két adatokkal feltöltött Contact példányt, és a print() függvénnyel kiírattam őket.

**T2** – Létrehoztam egy PhoneBook példányt (test), és a list\_contacts() függvénnyel kilistáztam az elemeit (amik itt még nem léteztek).

**T3** – Hozzáadtam az első Contact-ot a telefonkönyvhöz, majd kilistáztam az elemeit.

**T4** – Hozzáadtam a második Contact-ot is a telefonkönyvhöz, majd kilistáztam az elemeit.

**T5** – Ebben a tesztben felvettem egy Name példányt (Teszt Elek), majd a SearchByName() függvénnyel megkerestettem a telefonkönyvben, amit az meg is talált.

**T6** – Töröltem a tömbből az elsőként felvett Contact-ot, majd kiírattam a telefonkönyv tartalmát.

**T7** – Két újabb névjegyet vettem fel a kódban, és ezeket is hozzáadtam a telefonkönyvhöz és kilistáztam azt.

**T8** – Felvettem a kódban néhány próbaadatot, és ezeket megkerestettem a SearchBy…() függvényekkel. Hogy minden esetet bemutathassak, kettő valóban szereplő és kettő nem szereplő adatot adtam meg.

**T9** – A kódban felvett adatokra cseréltem le az egyik névjegy adatait a Change…() függvénnyel, aztán kilistáztam a módosított telefonkönyvet.

**T10** – A T9 esetben megváltoztatott nevet cseréljük le az InpName() függvény által beolvasott Name példányra, majd listázás.

**T11** – A kódban létrehoztam egy üres Contact-ot, hozzáadtam a telefonkönyvhöz, majd minden egyes adatát bekértem az Inp…() függvényekkel, majd kilistáztam a telefonkönyvet.

## **6. Tesztelés**

### **6.1 A tesztesetek eredményei**

**A képen képernyőkép látható

Automatikusan generált leírásA képen képernyőkép látható

Automatikusan generált leírásA képen képernyőkép látható

Automatikusan generált leírásA képen képernyőkép látható

Automatikusan generált leírás**Mellékeltem a tesztesetek elvárt kimenetét:

**A képen képernyőkép látható

Automatikusan generált leírásA képen képernyőkép látható

Automatikusan generált leírásA képen képernyőkép látható

Automatikusan generált leírás**

**A képen képernyőkép látható

Automatikusan generált leírásA képen képernyőkép látható

Automatikusan generált leírás**

**A képen képernyőkép látható

Automatikusan generált leírásA képen képernyőkép látható

Automatikusan generált leírás**

### **6.2 A memóriakezelés ellenőrzésének eredménye**

A memóriaszivárgást illetve hibás memória kezelést ellenőrző programok futása nem mutatott ki semmi féle rendellenességet, az elkészített program helyesen kezeli a saját memóriaterületét.

### **6.3 Lefedettségi teszt**

A lefedettségi teszt eredménye >95% lett, a kimaradó néhány részlet az egyes elágazások be nem következő ágait jelenti. A tesztesetek bővítésével ezt a számot lehetne tovább is növelni, de ezt én nem láttam szükségesnek.

## **7. Mellékletek**

Mellékeltem továbbá a beadott forráskódot, állományonkénti bontásban:

### **7.1 main.cpp**

#include <iostream>

#include "phonebook.hpp"

#include "memtrace.h"

int main**()**

**{**

String sep**(**"-----------------------------------------------------------\n"**);**

std**::**cout **<<** sep**;**

std**::**cout **<<** "Telefonkonyv OOP 2020/Prog2 - Toth Adam Laszlo / TK6NT3\n"**;**

std**::**cout **<<** sep **<<** std**::**endl**;**

std**::**cout **<<** sep**;**

std**::**cout **<<** "Az itt megvalositott tesztek kiirasaban a --- szaggatott vonal\na tesztesetek elkuloniteset szolgalja,mig a # jelzesek\nkozott egy-egy nevjegy tarolt adatai lathatoak.\n"**;**

std**::**cout **<<** "A keresesek erdemenyeit {} jelek,\na mig a telefonkonyv adott pillanatbeli tartalmat @ jelek\nzarjak kozre.\nA $$$ T tesztek leirasa a dokumentacioban megtalalhato.\n"**;**

std**::**cout **<<** sep **<<** "\n\n"**;**

std**::**cout **<<** sep**;**

//Telefonkonyv teszt, mukodes szemleltetese

Contact proba\_1**(**Name**(**"Elek"**,**"Teszt"**),**String**(**"Uttoro"**),**Address**(**"1111"**,**"Budapest"**,**"Ez utca"**,**"1"**),**PhNum**(**"123456"**,**"654321"**),**String**(**"telek@email.hu"**),**Date**(**1990**,**1**,**1**));**

Contact proba\_2**(**Name**(**"Janos"**,**"Kovacs"**),**String**(**"Tomi"**),**Address**(**"1122"**,**"Kecskemet"**,**"Fo utca"**,**"100"**),**PhNum**(**"11223344"**,**"44332211"**),**String**(**"kovajni@g.g"**),**Date**(**2000**,**10**,**23**));**

std**::**cout **<<** "$$$ T1 2 teszt contact letrehozva: \n"**;**

std**::**cout **<<** sep**;**

proba\_1**.**print**();**

proba\_2**.**print**();**

PhoneBook test**;**

std**::**cout **<<** sep**;**

std**::**cout **<<** "$$$ T2 Telefonkonyv peldany letrehozva\n"**;**

std**::**cout **<<** sep**;**

test**.**list\_contacts**();**

test**.**Add**(&**proba\_1**);**

std**::**cout **<<** sep**;**

std**::**cout **<<** "$$$ T3 Proba\_1 felveve a telefonkonyvbe\n"**;**

std**::**cout **<<** sep**;**

test**.**list\_contacts**();**

test**.**Add**(&**proba\_2**);**

std**::**cout **<<** sep**;**

std**::**cout **<<** "$$$ T4 Proba\_2 felveve, ezutan a telefonkonyv tartalma:\n"**;**

std**::**cout **<<** sep**;**

test**.**list\_contacts**();**

Name keres**(**"Elek"**,**"Teszt"**);**

std**::**cout **<<** sep**;**

std**::**cout **<<** "$$$ T5 Keressuk a kovetkezo nevet:\n"**;**

std**::**cout **<<** sep**;**

keres**.**print**();**

std**::**cout **<<** sep**;**

test**.**SearchByName**(**keres**);**

std**::**cout **<<** sep**;**

std**::**cout **<<** "$$$ T6 Toroljuk a kovetkezo contactot: \n"**;**

std**::**cout **<<** sep**;**

proba\_1**.**print**();**

std**::**cout **<<** sep**;**

std**::**cout **<<** "Ezutan a telefonkonyv tartalma: \n"**;**

std**::**cout **<<** sep**;**

test**.**Remove**(&**proba\_1**);**

test**.**list\_contacts**();**

Contact proba\_3**(**Name**(**"Adam"**,** "Toth"**),** String**(**"Adi"**),** Address**(**"1111"**,** "Budapest"**,** "Masik utca"**,** "3"**),** PhNum**(**"06060606060"**,** "-"**),** String**(**"cimem@email.hu"**),** Date**(**1999**,** 12**,** 12**));**

Contact proba\_4**(**Name**(**"Sandor"**,** "Szabo"**),** String**(**"-"**),** Address**(**"9999"**,** "Kistarcsa"**,** "Fo utca"**,** "10"**),** PhNum**(**"-"**,** "+36060606"**),** String**(**"sanyika@email.hu"**),** Date**(**1960**,** 10**,** 11**));**

test**.**Add**(&**proba\_3**);**

test**.**Add**(&**proba\_4**);**

std**::**cout **<<** sep**;**

std**::**cout **<<** "$$$ T7 Felvettunk meg 2 contactot,\nes ezeket is hozzadtuk a telefonkonyvhoz:\n"**;**

std**::**cout **<<** sep**;**

test**.**list\_contacts**();**

Name kname**(**"Pista"**,** "Kiss"**);**

Address kadd**(**"9999"**,** "Kistarcsa"**,** "Fo utca"**,** "10"**);**

PhNum knum**(**"06060606060"**,** "-"**);**

Date kdate**(**1999**,** 11**,** 20**);**

std**::**cout **<<** sep**;**

std**::**cout **<<** "$$$ T8 Felvettunk nehany proba adatot,\namiket megprobalunk megkeresni a telefonkonyvben:\n"**;**

std**::**cout **<<** sep**;**

std**::**cout **<<** sep**;**

std**::**cout **<<** "Ami alapjan keresunk:\n"**;**

kname**.**print**();**

std**::**cout **<<** sep**;**

test**.**SearchByName**(**kname**);**

std**::**cout **<<** sep**;**

std**::**cout **<<** "Ami alapjan keresunk:\n"**;**

kadd**.**print**();**

std**::**cout **<<** sep**;**

test**.**SearchByAddress**(**kadd**);**

std**::**cout **<<** sep**;**

std**::**cout **<<** "Ami alapjan keresunk:\n"**;**

knum**.**print**();**

std**::**cout **<<** sep**;**

test**.**SearchByNumber**(**knum**);**

std**::**cout **<<** sep**;**

std**::**cout **<<** "Ami alapjan keresunk:\n"**;**

kdate**.**print**();**

std**::**cout **<<** sep**;**

test**.**SearchByBirthday**(**kdate**);**

std**::**cout **<<** sep**;**

std**::**cout **<<** "$$$ T9 Felvettunk nehany proba adatot,\namiket megprobalunk lecserelni a telefonkonyvben:\n"**;**

std**::**cout **<<** sep**;**

Name valtozik**(**"Adam"**,** "Toth"**);**

Address ujc**(**"1111"**,** "Valahol"**,** "Barhol"**,** "0"**);**

PhNum ujs**(**"12121212"**,** "34343434"**);**

String ujn**(**"Pityuka"**);**

String ujm**(**"tadam@mail.hu"**);**

std**::**cout **<<** "A valtoztatni kivant nevjegyhez tartozo nev:\n"**;**

valtozik**.**print**();**

std**::**cout **<<** "Az uj adatok:\n"**;**

kname**.**print**();**

ujn**.**print**();**

ujc**.**print**();**

ujs**.**print**();**

ujm**.**print**();**

test**.**ChangeAddress**(**valtozik**,** ujc**);**

test**.**ChangeMail**(**valtozik**,** ujm**);**

test**.**ChangeNick**(**valtozik**,** ujn**);**

test**.**ChangeNumber**(**valtozik**,** ujs**);**

test**.**ChangeName**(**valtozik**,** kname**);**

std**::**cout **<<** sep**;**

std**::**cout **<<** "Valtoztatas utan kereses a regi nev szerint:\n"**;**

std**::**cout **<<** sep**;**

test**.**SearchByName**(**valtozik**);**

std**::**cout **<<** sep**;**

std**::**cout **<<** "A telefonkonyv mar az uj adatokat tartalmazza:\n"**;**

std**::**cout **<<** sep**;**

test**.**list\_contacts**();**

std**::**cout **<<** sep**;**

std**::**cout **<<** "$$$ T10 Most az elobb lecserelt nevet ujbol megvaltoztatjuk,\nde ezuttal beolvasott adatra:\n"**;**

std**::**cout **<<** sep**;**

test**.**ChangeName**(**kname**,** test**.**InpName**());**

test**.**list\_contacts**();**

std**::**cout **<<** sep**;**

std**::**cout **<<** "$$$ T11 Most pedig uj nevjegyet veszunk fel bemenetrol:\n"**;**

std**::**cout **<<** sep**;**

Contact uj**(**Name**(**""**,**""**),** String**(**""**),** Address**(**""**,**""**,**""**,**""**),** PhNum**(**""**,**""**),** String**(**""**),** Date**(**0**,**0**,**0**));**

test**.**Add**(&**uj**);**

test**.**ChangeName**(**Name**(**""**,**""**),** test**.**InpName**());**

test**.**ChangeNick**(**uj**.**GetName**(),** test**.**InpNick**());**

test**.**ChangeAddress**(**uj**.**GetName**(),** test**.**InpAddress**());**

test**.**ChangeNumber**(**uj**.**GetName**(),** test**.**InpNumber**());**

test**.**ChangeMail**(**uj**.**GetName**(),** test**.**InpEmail**());**

test**.**ChangeBD**(**uj**.**GetName**(),** test**.**InpBD**());**

std**::**cout **<<** sep**;**

test**.**list\_contacts**();**

std**::**cout **<<** sep **<<** "Ez a tesztesetek vege.\n" **<<** sep **<<**"Az oldal tetejen rovid magyarazat talalhato.\n" **<<** sep**;**

**}**

### **7.2 phonebook.cpp**

#include "phonebook.hpp"

#include "contact.hpp"

#include <iostream>

void PhoneBook**::**Add**(**Contact**\*** c**)**

**{**

entry**.**bovit**(**c**);**

**}**

void PhoneBook**::**Remove**(**Contact**\*** c**)**

**{**

**if(**entry**.**GetLength**()** **==** 0**)** **return;**

entry**.**elem\_ki**(**c**);**

**}**

void PhoneBook**::**list\_contacts**()**

**{**

std**::**cout **<<** "@Telefonkonyv kezdodik\n"**;**

**for(**size\_t i **=** 0**;** i**<** entry**.**GetLength**();**i**++)**

**{**

entry**[**i**]->**print**();**

**}**

**if** **(**entry**.**GetLength**()** **==** 0**)** std**::**cout **<<** "A telefonkonyv ures!\n"**;**

std**::**cout **<<** "@Telefonkonyv vege\n"**;**

**}**

void PhoneBook**::**SearchByName**(**const Name**&** n**)**

**{**

std**::**cout **<<** "{ Kereses eleje\n\n"**;**

**for** **(**size\_t i **=** 0**;** i **<** entry**.**GetLength**();** i**++)**

**{**

**if** **(**entry**[**i**]->**GetName**()** **==** n**)**

**{**

std**::**cout **<<** "Sikeres kereses! Talalat:\n"**;**

entry**[**i**]->**print**();**

std**::**cout **<<** "} Kereses vege\n"**;**

**return;**

**}**

**}**

std**::**cout **<<** "Nincs talalat\n\n"**;**

std**::**cout **<<** "} Kereses vege\n"**;**

**}**

void PhoneBook**::**SearchByBirthday**(**const Date bd**)**

**{**

std**::**cout **<<** "{ Kereses eleje\n\n"**;**

**for** **(**size\_t i **=** 0**;** i **<** entry**.**GetLength**();** i**++)**

**{**

**if** **(**entry**[**i**]->**GetBD**()** **==** bd**)**

**{**

std**::**cout **<<** "Sikeres kereses! Talalat:\n"**;**

entry**[**i**]->**print**();**

std**::**cout **<<** "} Kereses vege\n"**;**

**return;**

**}**

**}**

std**::**cout **<<** "Nincs talalat\n\n"**;**

std**::**cout **<<** "} Kereses vege\n"**;**

**}**

void PhoneBook**::**SearchByAddress**(**const Address**&** a**)**

**{**

std**::**cout **<<** "{ Kereses eleje\n\n"**;**

**for** **(**size\_t i **=** 0**;** i **<** entry**.**GetLength**();** i**++)**

**{**

**if** **(**entry**[**i**]->**GetAddress**()** **==** a**)**

**{**

std**::**cout **<<** "Sikeres kereses! Talalat:\n"**;**

entry**[**i**]->**print**();**

std**::**cout **<<** "} Kereses vege\n"**;**

**return;**

**}**

**}**

std**::**cout **<<** "Nincs talalat\n\n"**;**

std**::**cout **<<** "} Kereses vege\n"**;**

**}**

void PhoneBook**::**SearchByNumber**(**const PhNum**&** num**)**

**{**

std**::**cout **<<** "{ Kereses eleje\n\n"**;**

**for** **(**size\_t i **=** 0**;** i **<** entry**.**GetLength**();** i**++)**

**{**

**if** **(**entry**[**i**]->**GetNumber**()** **==** num**)**

**{**

std**::**cout **<<** "Sikeres kereses! Talalat:\n"**;**

entry**[**i**]->**print**();**

std**::**cout **<<** "} Kereses vege\n"**;**

**return;**

**}**

**}**

std**::**cout **<<** "Nincs talalat\n\n"**;**

std**::**cout **<<** "} Kereses vege\n"**;**

**}**

void PhoneBook**::**ChangeName**(**const Name**&** n**,** const Name**&** m**)**

**{**

**for** **(**size\_t i **=** 0**;** i **<** entry**.**GetLength**();** i**++)**

**{**

**if** **(**entry**[**i**]->**GetName**()** **==** n**)**

**{**

entry**[**i**]->**SetName**(**m**);**

std**::**cout **<<** "Sikeres nevvaltoztatas.\n"**;**

**return;**

**}**

**}**

std**::**cout **<<** "Nem talaltam ilyen nevet a telefonkonyvben.\n"**;**

**}**

void PhoneBook**::**ChangeNumber**(**const Name**&** n**,** const PhNum**&** m**)**

**{**

**for** **(**size\_t i **=** 0**;** i **<** entry**.**GetLength**();** i**++)**

**{**

**if** **(**entry**[**i**]->**GetName**()** **==** n**)**

**{**

entry**[**i**]->**SetPhNum**(**m**);**

std**::**cout **<<** "Sikeres szamvaltoztatas.\n"**;**

**return;**

**}**

**}**

std**::**cout **<<** "Nem talaltam ilyen nevet a telefonkonyvben.\n"**;**

**}**

void PhoneBook**::**ChangeAddress**(**const Name**&** n**,** const Address**&** m**)**

**{**

**for** **(**size\_t i **=** 0**;** i **<** entry**.**GetLength**();** i**++)**

**{**

**if** **(**entry**[**i**]->**GetName**()** **==** n**)**

**{**

entry**[**i**]->**SetAddress**(**m**);**

std**::**cout **<<** "Sikeres cimvaltoztatas.\n"**;**

**return;**

**}**

**}**

std**::**cout **<<** "Nem talaltam ilyen nevet a telefonkonyvben.\n"**;**

**}**

void PhoneBook**::**ChangeNick**(**const Name**&** n**,** const String**&** m**)**

**{**

**for** **(**size\_t i **=** 0**;** i **<** entry**.**GetLength**();** i**++)**

**{**

**if** **(**entry**[**i**]->**GetName**()** **==** n**)**

**{**

entry**[**i**]->**SetNick**(**m**);**

std**::**cout **<<** "Sikeres nicknevvaltoztatas.\n"**;**

**return;**

**}**

**}**

std**::**cout **<<** "Nem talaltam ilyen nevet a telefonkonyvben.\n"**;**

**}**

void PhoneBook**::**ChangeMail**(**const Name**&** n**,** const String**&** m**)**

**{**

**for** **(**size\_t i **=** 0**;** i **<** entry**.**GetLength**();** i**++)**

**{**

**if** **(**entry**[**i**]->**GetName**()** **==** n**)**

**{**

entry**[**i**]->**SetMail**(**m**);**

std**::**cout **<<** "Sikeres e-cimvaltoztatas.\n"**;**

**return;**

**}**

**}**

std**::**cout **<<** "Nem talaltam ilyen nevet a telefonkonyvben.\n"**;**

**}**

void PhoneBook**::**ChangeBD**(**const Name**&** n**,** const Date**&** m**)**

**{**

**for** **(**size\_t i **=** 0**;** i **<** entry**.**GetLength**();** i**++)**

**{**

**if** **(**entry**[**i**]->**GetName**()** **==** n**)**

**{**

entry**[**i**]->**SetDate**(**m**);**

std**::**cout **<<** "Sikeres datumvaltoztatas.\n"**;**

**return;**

**}**

**}**

std**::**cout **<<** "Nem talaltam ilyen nevet a telefonkonyvben.\n"**;**

**}**

Name PhoneBook**::**InpName**()**

**{**

String fn**;**

String ln**;**

std**::**cout **<<** "Please type in the firstname:\n"**;**

std**::**cin **>>** fn**;**

std**::**cout **<<** "Please type in the lastname:\n"**;**

std**::**cin **>>** ln**;**

**return** Name**(**fn**,** ln**);**

**}**

String PhoneBook**::**InpNick**()**

**{**

String nick**;**

std**::**cout **<<** "Please type in the nickname:\n"**;**

std**::**cin **>>** nick**;**

**return** nick**;**

**}**

Address PhoneBook**::**InpAddress**()**

**{**

String zip**;**

String ct**;**

String st**;**

String hn**;**

std**::**cout **<<** "Please type in the ZIP code:\n"**;**

std**::**cin **>>** zip**;**

std**::**cout **<<** "Please type in the City:\n"**;**

std**::**cin **>>** ct**;**

std**::**cout **<<** "Please type in the Street name:\n"**;**

std**::**cin **>>** st**;**

std**::**cout **<<** "Please type in the House number:\n"**;**

std**::**cin **>>** hn**;**

**return** Address**(**zip**,** ct**,** st**,** hn**);**

**}**

PhNum PhoneBook**::**InpNumber**()**

**{**

String w**;**

String m**;**

std**::**cout **<<** "Please type in the Work number:\n"**;**

std**::**cin **>>** w**;**

std**::**cout **<<** "Please type in the Mobile number:\n"**;**

std**::**cin **>>** m**;**

**return** PhNum**(**w**,** m**);**

**}**

String PhoneBook**::**InpEmail**()**

**{**

String email**;**

std**::**cout **<<** "Please type in the Email address:\n"**;**

std**::**cin **>>** email**;**

**return** email**;**

**}**

Date PhoneBook**::**InpBD**()**

**{**

int y**;**

int m**;**

int d**;**

std**::**cout **<<** "Please type in the year:\n"**;**

std**::**cin **>>** y**;**

std**::**cout **<<** "Please type in the month:\n"**;**

std**::**cin **>>** m**;**

std**::**cout **<<** "Please type in the day:\n"**;**

std**::**cin **>>** d**;**

**return** Date**(**y**,** m**,** d**);**

**}**

### **7.3 phonebook.hpp**

#ifndef PHONEBOOK\_HPP

#define PHONEBOOK\_HPP

#include "array.hpp"

#include "contact.hpp"

#include <iostream>

class PhoneBook

**{**

Array**<**Contact**\*>** entry**;**

public**:**

void list\_contacts**();**

void Add**(**Contact**\*** c**);**

void Remove**(**Contact**\*** c**);**

void SearchByName**(**const Name**&** n**);**

void SearchByNumber**(**const PhNum**&** num**);**

void SearchByAddress**(**const Address**&** a**);**

void SearchByBirthday**(**const Date bd**);**

void ChangeName**(**const Name**&** n**,** const Name**&** m**);**

void ChangeNumber**(**const Name**&** n**,** const PhNum**&** m**);**

void ChangeAddress**(**const Name**&** n**,** const Address**&** m**);**

void ChangeMail**(**const Name**&** n**,** const String**&** m**);**

void ChangeNick**(**const Name**&** n**,** const String**&** m**);**

void ChangeBD**(**const Name**&** n**,** const Date**&** m**);**

Name InpName**();**

String InpNick**();**

Address InpAddress**();**

PhNum InpNumber**();**

String InpEmail**();**

Date InpBD**();**

**};**

#endif // PHONEBOOK\_HPP

### **7.4 contact.hpp**

#ifndef CONTACT\_HPP

#define CONTACT\_HPP

#include <iostream>

#include "string.h"

#include "stored\_types.hpp"

class Contact

**{**

Name name**;**

String nickname**;**

Address address**;**

PhNum number**;**

String email**;**

Date birthdate**;**

public**:**

Contact**(**Name n**,** String nick**,**Address a**,**PhNum num**,**String e**,**Date d**):** name**(**n**),**nickname**(**nick**),**address**(**a**),**number**(**num**),**email**(**e**),**birthdate**(**d**){};**

virtual void print**()**

**{**

std**::**cout **<<** "#Nevjegy eleje\n|\n"**;**

std**::**cout **<<** "| "**;** name**.**print**();**

std**::**cout **<<** "| "**;** std**::**cout **<<** "Becenev:\n|\t\t" **<<** nickname **<<** "\n"**;**

std**::**cout **<<** "| "**;** address**.**print**();**

std**::**cout **<<** "| "**;** number**.**print**();**

std**::**cout **<<** "| "**;** std**::**cout **<<** "Email:\n|\t\t" **<<** email **<<** "\n"**;**

std**::**cout **<<** "| "**;** birthdate**.**print**();**

std**::**cout **<<** "#Nevjegy vege\n"**;**

std**::**cout **<<** std**::**endl**;**

**}**

const Name GetName**(){return** name**;}**

const String GetNick**(){return** nickname**;}**

const Address GetAddress**(){return** address**;}**

const PhNum GetNumber**(){return** number**;}**

const String GetEmail**(){return** email**;}**

const Date GetBD**(){return** birthdate**;}**

void SetName**(**const Name**&** n**)** **{** name **=** n**;}**

void SetNick**(**const String**&** n**)** **{** nickname **=** n**;** **}**

void SetAddress**(**const Address**&** n**)** **{** address **=** n**;** **}**

void SetPhNum**(**const PhNum**&** n**)** **{** number **=** n**;** **}**

void SetMail**(**const String**&** n**)** **{** email **=** n**;** **}**

void SetDate**(**const Date**&** n**)** **{** birthdate **=** n**;** **}**

virtual **~**Contact**()** **{}**

**};**

#endif // CONTACT\_HPP

### **7.5 array.hpp**

#ifndef ARRAY\_HPP

#define ARRAY\_HPP

#include <stddef.h>

#include "contact.hpp"

template **<**class T**>**

class Array

**{**

protected**:**

T**\*** Elements**;**

size\_t length**;**

public**:**

Array**(**size\_t h **=** 0**)** **:**length**(**h**){**Elements **=** **new** T**[**length**];}**

Array**(**Array**<**T**>&** o**)**

**{**

length **=** o**.**length**;**

Elements **=** **new** T**[**length**];**

**for(**size\_t i **=** 0**;**i**<**length**;**i**++)**

**{**

Elements**[**i**]** **=** o**.**Elements**[**i**];**

**}**

**}**

T**&** **operator[](**size\_t n**){return** Elements**[**n**];}**

Array**&** **operator=(**const Array**&** c**)**

**{**

**if(\*this** **!=** c**)**

**{**

length **=** c**.**length**;**

**delete[]** Elements**;**

**for(**size\_t i **=** 0**;**i**<**length**;**i**++)**

**{**

Elements**[**i**]** **=** c**.**Elements**[**i**];**

**}**

**}**

**return** **this;**

**}**

T**\*** GetElement**(){return** Elements**;}**

size\_t GetLength**(){return** length**;}**

**~**Array**(){delete[]** Elements**;}**

void bovit**(**Contact**\*** be**)**

**{**

**for(**size\_t i **=** 0**;**i**<**length**;**i**++)**

**{**

**if(**Elements**[**i**]** **==** be**)** **return;**

**}**

length**++;**

T**\*** uj **=** **new** T**[**length**];**

**for(**size\_t i **=** 0**;**i**<**length**-**1**;**i**++)**

**{**

uj**[**i**]** **=** Elements**[**i**];**

**}**

**delete[]** Elements**;**

uj**[**length**-**1**]** **=** be**;**

Elements **=** uj**;**

**}**

void elem\_ki**(**Contact**\*** ki**)**

**{**

bool eleme **=** **false;**

**for(**size\_t i **=** 0**;**i**<**length**;**i**++)**

**{**

**if(**Elements**[**i**]** **==** ki**)** eleme **=** **true;**

**}**

**if(**eleme **==** 0**)** **return;**

length**--;**

T**\*** uj **=** **new** T**[**length**];**

size\_t j **=** 0**;**

**for(**size\_t i **=** 0**;**i**<**length**+**1**;**i**++)**

**{**

**if(**Elements**[**i**]** **!=** ki**){** uj**[**j**]** **=** Elements**[**i**];**j**++;}**

**}**

**delete[]** Elements**;**

Elements **=** uj**;**

**}**

**};**

#endif // ARRAY\_HPP

### **7.6 stored\_types.hpp**

#ifndef STORED\_TYPES\_HPP

#define STORED\_TYPES\_HPP

#include <iostream>

#include "string.h"

class Name //Vezeteknev es keresztnev tarolasa

**{**

protected**:**

String firstName**;**

String lastName**;**

public**:**

Name**(**const char**\*** first**,**const char**\*** last**):** firstName**(**first**),**lastName**(**last**){};**

Name**(**String first**,** String last**)** **:** firstName**(**first**),** lastName**(**last**)** **{};**

String GetFN**()** const **{return** firstName**;}**

String GetLN**()** const **{return** lastName**;}**

String GetFullName**()** const **{return** firstName **+** " " **+** lastName**;}**

void print**(){**std**::**cout **<<** "Teljes nev:\n| \t\t" **<<** GetFullName**()** **<<** "\n"**;}**

friend bool **operator==(**const Name n**,** const Name m**)**

**{**

**if** **(**n**.**firstName **==** m**.**firstName **&&** n**.**lastName **==** m**.**lastName**)**

**{**

**return** **true;**

**}**

**else** **return** **false;**

**}**

virtual **~**Name**()** **{}**

**};**

class PhNum //Munkahelyi es mobiltelefonszam tarolasa

**{**

protected**:**

String workTel**;**

String mobile**;**

public**:**

PhNum**(**const char**\*** wt**,** const char**\*** m**):** workTel**(**wt**),**mobile**(**m**){};**

PhNum**(**String wt**,** String m**)** **:** workTel**(**wt**),** mobile**(**m**)** **{};**

String GetWN**(){return** workTel**;}**

String GetMobile**(){return** mobile**;}**

void print**(){**std**::**cout **<<** "Munkah. szam:\n| \t\t" **<<** GetWN**()** **<<** "\n| Mobilszam:\n| \t\t" **<<** GetMobile**()** **<<** "\n"**;}**

friend bool **operator==(**const PhNum n**,** const PhNum m**)**

**{**

**if** **(**n**.**workTel **==** m**.**workTel **||** n**.**mobile **==** m**.**mobile**)**

**{**

**return** **true;**

**}**

**else** **return** **false;**

**}**

virtual **~**PhNum**()** **{}**

**};**

class Address //Lakcim tarolasa(irsz,varos,utca,hazszam)

**{**

protected**:**

String zip**;**

String city**;**

String street**;**

String hnum**;**

public**:**

Address**(**const char**\*** z**,** const char**\*** c**,**const char**\*** s**,**const char**\*** n**)** **:**zip**(**z**),**city**(**c**),**street**(**s**),**hnum**(**n**){};**

Address**(**String z**,** String c**,** String s**,** String n**)** **:**zip**(**z**),** city**(**c**),** street**(**s**),** hnum**(**n**)** **{};**

String GetZIP**(){return** zip**;}**

String GetCity**(){return** city**;}**

String GetAddress**(){return** street **+** " " **+** hnum**;}**

void print**(){**std**::**cout **<<** "Lakcim:\n| \t\t" **<<** GetZIP**()** **+** " " **+** GetCity**()** **+** " " **+** GetAddress**()** **+** "\n"**;}**

friend bool **operator==(**const Address a1**,** const Address a2**)**

**{**

**if** **(**a1**.**zip **==** a2**.**zip **&&** a1**.**city **==** a2**.**city **&&** a1**.**street **==** a2**.**street **&&** a1**.**hnum **==** a2**.**hnum**)**

**{**

**return** **true;**

**}**

**else** **return** **false;**

**}**

virtual **~**Address**()** **{}**

**};**

class Date //Datum eltarolasa (szuletesnaphoz)

**{**

protected**:**

int year**;**

int month**;**

int day**;**

public**:**

Date**(**int y**,**int m**,**int d**):** year**(**y**),**month**(**m**),**day**(**d**){};**

int GetYear**(){return** year**;}**

int GetMonth**(){return** month**;}**

int GetDay**(){return** day**;}**

void SetDate**(**int y**,** int m**,** int d**)** **{** year **=** y**;** month **=** m**;** day **=** d**;}**

void print**(){**std**::**cout **<<** "Szul. datum:\n| \t\t"**<<** GetYear**()** **<<** "." **<<** GetMonth**()** **<<** "." **<<** GetDay**()** **<<** "\n"**;}**

friend bool **operator==(**const Date n**,** const Date m**)**

**{**

**if** **(**n**.**year **==** m**.**year **&&** n**.**month **==** m**.**month **&&** n**.**day **==** m**.**day**)**

**{**

**return** **true;**

**}**

**else** **return** **false;**

**}**

virtual **~**Date**()** **{}**

**};**

#endif // STORED\_TYPES\_HPP

### **7.7 string.cpp**

#include <iostream> // Kiíratáshoz

#include <cstring> // Sztringműveletekhez

#include "string.h"

**using** std**::**ios\_base**;**

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

**{**

len **=** 1**;**

pData **=** **new** char**[**len**+**1**];**

pData**[**0**]** **=** ch**;**

pData**[**1**]** **=** '\0'**;**

**}**

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

**{**

len **=** strlen**(**p**);**

pData **=** **new** char**[**len**+**1**];**

strcpy**(**pData**,** p**);**

**}**

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

**{**

len **=** s1**.**len**;**

pData **=** **new** char**[**len**+**1**];**

strcpy**(**pData**,** s1**.**pData**);**

**}**

String**::~**String**()**

**{**

**delete[]** pData**;**

**}**

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;**

**}**

char**&** String**::operator[](**unsigned int idx**)**

**{**

**if** **(**idx **>=** len**)** **throw** "ERROR 404"**;**

**return** pData**[**idx**];**

**}**

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**;**

**}**

bool **operator** **==** **(**const String**&** lhs**,** const String**&** rhs**)**

**{**

**return** strcmp**(**lhs**.**pData**,** rhs**.**pData**)** **==** 0**;**

**}**

std**::**ostream**&** **operator<<(**std**::**ostream**&** os**,** const String**&** s0**)**

**{**

os **<<** s0**.**c\_str**();**

**return** os**;**

**}**

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

unsigned char ch**;**

s0 **=** String**(**""**);**

std**::**ios\_base**::**fmtflags fl **=** is**.**flags**();**

is**.**setf**(**ios\_base**::**skipws**);**

**while** **(**is **>>** ch**)** **{**

is**.**unsetf**(**ios\_base**::**skipws**);**

**if** **(**isspace**(**ch**))** **{**

is**.**putback**(**ch**);**

**break;**

**}**

**else** **{**

s0 **=** s0 **+** ch**;**

**}**

**}**

is**.**setf**(**fl**);**

**return** is**;**

**}**

### **7.8 string.h**

#ifndef STRING\_H

#define STRING\_H

#include <iostream>

class String

**{**

char **\***pData**;**

size\_t len**;**

public**:**

void printDbg**(**const char **\***txt **=** ""**)** const

**{**

std**::**cout **<<** txt **<<** "[" **<<** len **<<** "], "

**<<** **(**pData **?** pData **:** "(NULL)"**)** **<<** '|' **<<** std**::**endl**;**

**}**

String**()** **:**pData**(**0**),** len**(**0**)** **{}**

size\_t size**()** const

**{**

**return** len**;**

**}**

const char**\*** c\_str**()** const

**{**

**if** **(**pData **==** **NULL)** **return** ""**;**

**else** **return** pData**;**

**}**

virtual **~**String**();**

String**(**char ch**);**

String**(**const char **\***p**);**

String**(**const String**&** s1**);**

void print**()** **{** std**::**cout **<<** pData **<<** std**::**endl**;** **}**

String**&** **operator=(**const String**&** rhs**);**

friend bool **operator==** **(**const String**&** lhs**,** const String**&** rhs**);**

String **operator+(**const String**&** rhs**)** const **;**

char**&** **operator[](**unsigned int idx**);**

**};**

std**::**ostream**&** **operator<<(**std**::**ostream**&** os**,** const String**&** s0**);**

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

#endif