# Végleges dokumentáció

NHF1 – Specifikáció:

# Mobilszolgáltatás nyilvántartó

A választott feladat egy olyan nyilvántartáskezelő program, melyben egy mobilszolgáltató cég ügyfeleinek az egyes adatait lehet kezelni. A szolgáltató cég rendszere az ügyfeleiknek a nevét, címét és a telefonszámát tárolja (ami egyben az egyéni azonosítószáma is), melyekhez hozzárendeli az ügyfél saját szolgáltatáscsomagját. Valamint a program számontartja az ügyfelek az adott hónapban küldött SMS-eit és lebeszélt perceit.

A program elindításakor egy menürendszer tárul elénk melyben lehetőségünk lesz a következőkre:

- új ügyfelet felvenni
- meglévő ügyfelet törölni a rendszerből
- kilistázni a rendszerben szereplő ügyfelek adatait
- módosítani a bármelyik ügyfél adatait
- kilépni a programból

Az ügyfelek neve, címe és telefonszáma, valamint a küldött SMS-ek és lebeszélt percek két külön fájlban kerülnek majd tárolásra. A program listázáskor kiírja, hogy az egy adott ügyfélnek mennyit kell majd fizetnie a forgalom napján. Az ügyfelek a rendszerben a telefonszámaik (azaz egyéni azonosítójuk) alapján lesznek sorba rendezve, ezzel segítve az adatok könnyebb kezelését.

NHF2 – Terv:

# Terv

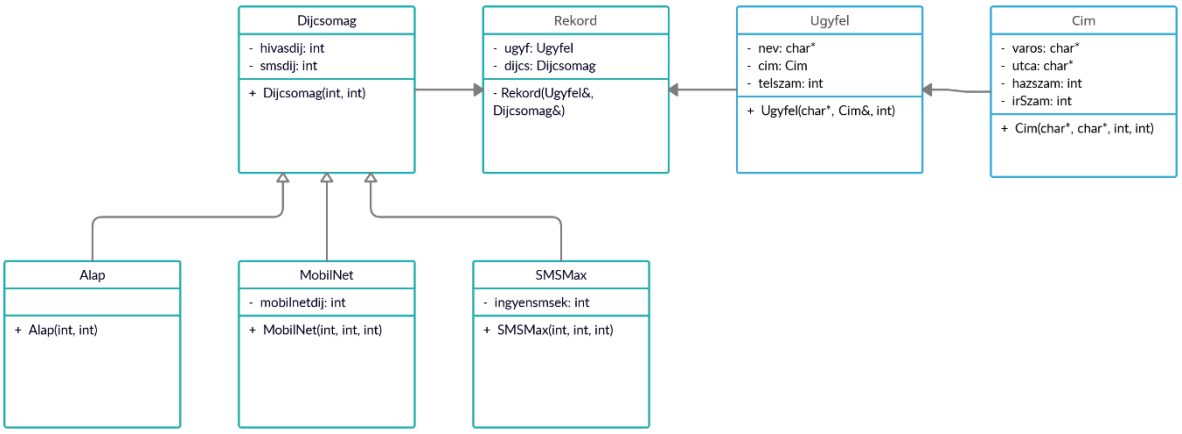
A program elindításakor egy menürendszer tárul elénk melyben lehetőségünk lesz a következőkre:

- új ügyfelet felvenni
  - ezt a menüpontot választva megadhatjuk az új ügyfél nevét, címét és telefonszámát (egyéni azonosítószám), majd elmenthetjük a már meglévő ügyfélrekordokhoz
- meglévő ügyfelet törölni a rendszerből
  - ebben a menüpontban egy adott ügyfél azonosítójának (telefonszám) a megadásával törölhetjük őt a rekordok közül
- kilistázni a rendszerben szereplő ügyfelek adatait
  - itt megnézhetjük egyben a rendszerbe felvett összes ügyfelet és a hozzájuk tartozó adataikat
- módosítani a bármelyik ügyfél adatait
  - ezen opciónál lehetőségünk nyílik egy adott ügyfél bármelyik adatát módosítani az adott ügyfél azonosítószáma megadása után
- kilépni a programból

Az ügyfelek neve, címe, telefonszáma és a szolgáltatáscsomagja, valamint a küldött SMS-ek és lebeszélt percek két külön fájlban kerülnek majd tárolásra. A program listázáskor kiírja, hogy az egy adott ügyfélnek mennyit kell majd fizetnie a forgalom napján. Az ügyfelek a rendszerben a telefonszámaik (azaz egyéni azonosítójuk) alapján lesznek sorba rendezve, ezzel segítve az adatok könnyebb kezelését.

A mobilszolgáltató három darab szolgáltatáscsomaggal rendelkezik: Alap, MobiNet és SMSMax. Minden csomag rendelkezik a mega előnyeivel és hátrányaival, így az ügyfelek maguk dönthetik el melyik csomag a számukra legmegfelelőbb.

A feladat megoldásához készített osztálydiagram:



NHF3 – Skeleton:

# Terv

A program elindításakor egy menürendszer tárul elénk melyben lehetőségünk lesz a következőkre:

- új ügyfelet felvenni
    - ezt a menüpontot választva megadhatjuk az új ügyfél nevét, címét és telefonszámát (egyéni azonosítószám), majd elmenthetjük a már meglévő ügyfélrekordokhoz

- meglévő ügyfelet törölni a rendszerből
    - ebben a menüpontban egy adott ügyfél azonosítójának (telefonszám) a megadásával törölhetjük őt a rekordok közül

- kilistázni a rendszerben szereplő ügyfelek adatait
    - itt megnézhetjük egyben a rendszerbe felvett összes ügyfelet és a hozzájuk tartozó adataikat

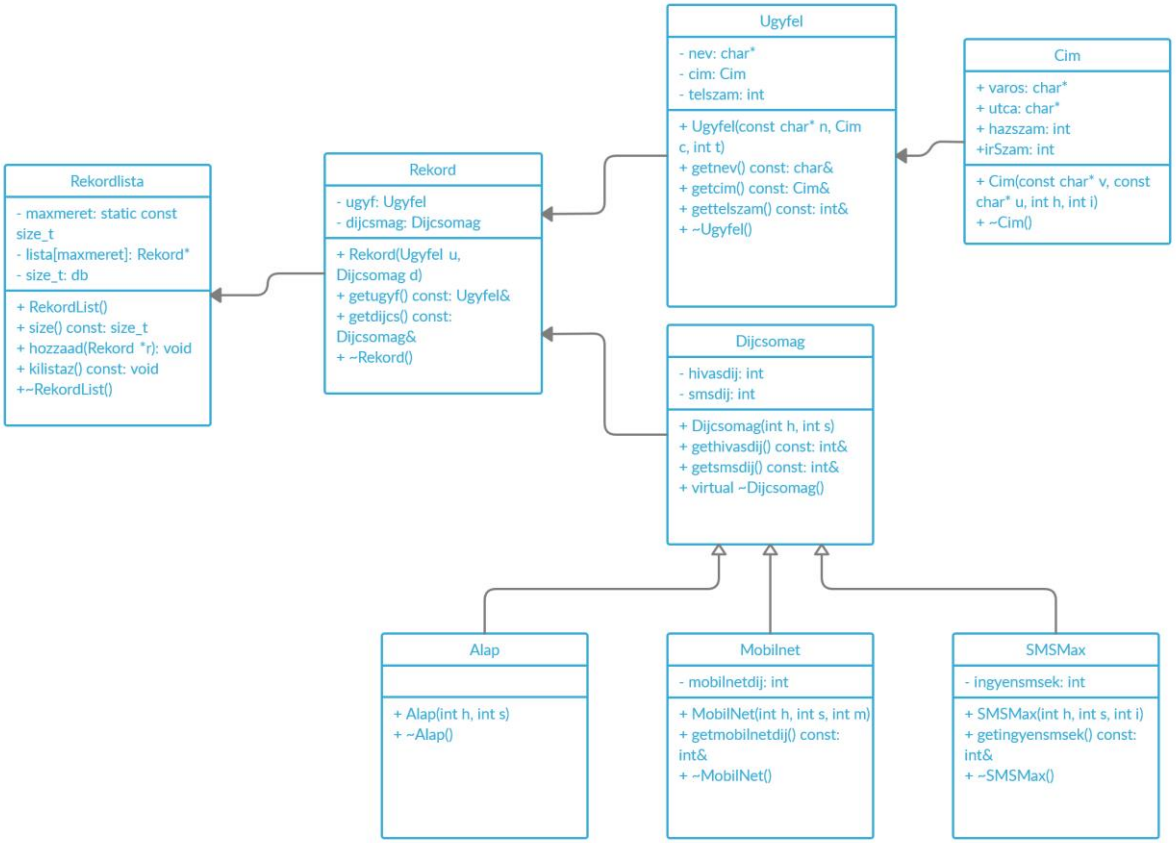
- módosítani a bármelyik ügyfél adatait
    - ezen opciónál lehetőségünk nyílik egy adott ügyfél bármelyik adatát módosítani az adott ügyfél azonosítószáma megadása után

- kilépni a programból

Az ügyfelek neve, címe, telefonszáma és a szolgáltatáscsomagja, valamint a küldött SMS-ek és lebeszélt percek két külön fájlban kerülnek majd tárolásra. A program listázáskor kiírja, hogy az egy adott ügyfélnek mennyit kell majd fizetnie a forgalom napján. Az ügyfelek a rendszerben a telefonszámaik (azaz egyéni azonosítójuk) alapján lesznek sorba rendezve, ezzel segítve az adatok könnyebb kezelését.

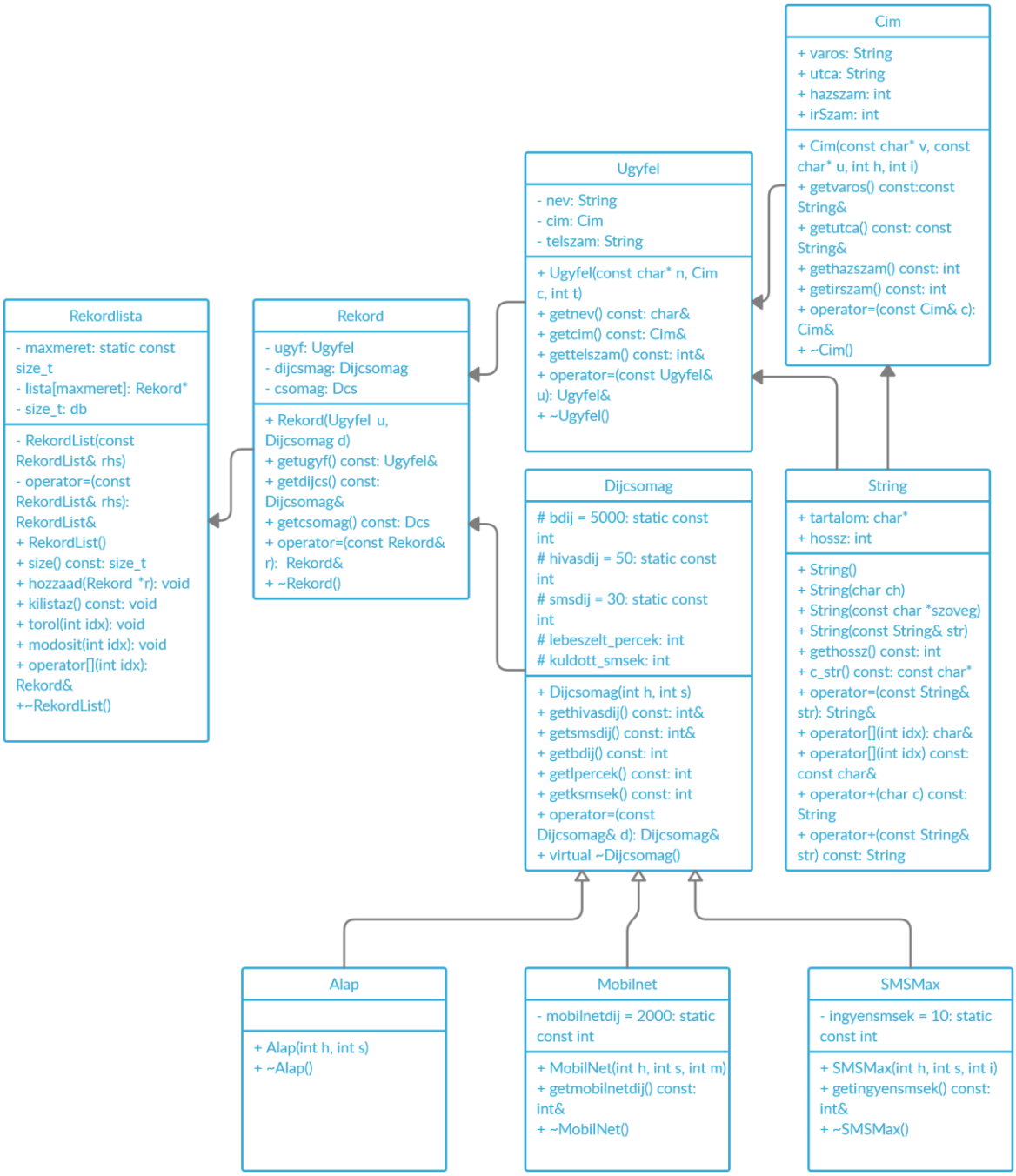
A mobilszolgáltató három darab szolgáltatáscsomaggal rendelkezik: Alap, MobiNet és SMSMax. Minden csomag rendelkezik a mega előnyeivel és hátrányaival, így az ügyfelek maguk dönthetik el melyik csomag a számukra legmegfelelőbb.

A feladat megoldásához készített osztálydiagram:



## NHF4 – Végleges specifikáció:

### Végleges osztálydiagram:



Változtatások az előző specifikációkhoz képest:

- Új osztály került hozzáadásra, egy sajaát készítésű String osztály
- Az ügyfelek nem a telefonszámuk alapján lettek sorbarendezve, hanem minden rekord kapott egy saját indexet, mely szerint lehet hivatkozni más funkciókban egy adott rekordra

# Osztályok bemutatása:

## Cim osztály:

adattagok:

- String varos: String típusú változó melyben a váosevet tárolom
- String utca: String típusú változó melyben az utca nevét tárolom
- int hazszam: int típusú változó melyben a házszámot tárolom
- int irSzam: int típusú változó melyben a irányítószámot tárolom

tagfüggvények:

- Cim(const String& v, const String& u, int h, int i): Konstruktor
- const String& getvaros() const: Visszaadja a varos nevû változót
- const String& getutca() const: Visszaadja az utca nevû változót
- int gethazszam() const: Visszaadja a hazszam nevû változót
- int getirszam() const: Visszaadja a irSzam nevû változót
- Cim& operator=(const Cim& c): = operátor, Cim típusú adat elemeit másolja bele a változókba
- ~Cim(): Destruktor

## String osztály:

adattagok:

- char *tartalom: Ebben a char* típusú változóban tároljuk el a string tartalmát
- int hossz: Ebben az int típusú változóban tárolom el a string hosszát

tagfüggvények:

- String(): Paraméter nélküli konstruktor
- String(char ch): Konstruktor, karaterből készít stringet
- String(const char *szoveg): Char*-ból készít string típusú változót
- String(const String& str): Másoló konstruktor: String-bõl készít (createString)
- int gethossz() const: Sztring hosszát adja vissza
- const char* c_str() const: C-sztringet ad vissza
- String& operator=(const String& str): Értékadó operátor
- char& operator[](int idx): Index operátor
- const char& operator[](int idx) const: Index operátor
- String operator+(char c) const: + operator, egy karatert ad a meglévő string-hez
- String operator+(const String& str) const: + operátor, ami a String-hez jobbról karaktert ad
- ~String(): Destruktor

## Dijcsomag osztály:

adattagok:

- static const int bdij = 5000: Az alap dij amit az ügyfeleknek fizetnie kell bármelyik díjcsomagért
- static const int hivasdij = 50: Telefonálás percenkénti díja
- static const int smsdij = 30: SMS küldés percenkénti díja
- int lebeszelt_percek: Egy adott ügyfél által lebeszélt perceknek a száma
- int kuldott_smsek: Egy adott ügyfél által küldött SMS-ek a száma

tagfüggvények:

- Dijcsomag(int h, int s): Konstruktor
- int gethivasdij() const: Visszaadja a hivasdij értékét
- int getsmsdij() const: Visszaadja a smsdij értékét
- int getbdij() const: Visszaadja a bdij értékét
- int getlpercek() const: Visszaadja a lebeszelt_percek számát
- int getksmsek() const: Visszaadja a kuldott_smsek számát
- Dijcsomag& operator=(const Dijcsomag& d): = operátor
- virtual ~Dijcsomag(): Destruktor, vitruális, hogy a leszármazottak is tudják használni

## Alap osztály:

adattagok:

-

tagfüggvények:

- Alap(int h, int s): Konstruktor
- ~Alap(): Destruktor

## MobilNet osztály:

adattagok:

- static const int mobilnetdij = 2000: Pulsz dij amit az ügyfélnke ki kell fizetnie, ha ezt a dijcsomagot választja

tagfüggvények:

- MobilNet(int h, int s): Konstruktor
- int getmobilnetdij() const: Visszaadja a mobilnetdij értékét
- ~MobilNet(): Destruktor

## SMSMax osztály:

adattagok:

- static const int ingyensmsek = 10: Ebben a változóban eltárolt szám adja meg, hogy mennyi ingyen sms-t tud küldeni az ügyfél

tagfüggvények:

- SMSMax(int h, int s): Konstruktor
- int getingyensmsek() const: Visszaadja a ingyensmsek értékét
- ~SMSMax(): Destruktor

## Ugyfel osztály:

adattagok:

- String nev: Az ügyfél neve String típusú változóban eltárolva
- Cim cím: Az ügyfél címe Cim típusú változóban eltárolva
- String telszam: Az ügyfél telefonszáma String típusú változóban eltárolva

tagfüggvények:

- Ugyfel(const String& n, Cim c, const String& t): Konstruktor
- const String& getnev() const: Visszaadja a nev valtozo tartalmat
- const Cim& getcim() const: Visszaadja a cim valtozo tartalmat
- const String& gettelszam() const: Visszaadja a telszam valtozo tartalmat
- Ugyfel& operator=(const Ugyfel& u): = operator
- ~Ugyfel(): Destruktor

## Rekord osztály:

adattagok:

- Ugyfel ugyf: Ugyfel típusú változóban tárolom az ügyfél szeméélyes adatait
- Dijcsomag dijcs: Dijcsomag típusú változóban tárolom az ügyfél dijcsomagával kapcsolatos adatokat
- Dcs csomag: Dcs enum típusú változóban tárolom el, hogy egy ügyfélnek milyenfajta dijcsomaggal rendelkezik

tagfüggvények:

- Rekord(Ugyfel u, Dijcsomag d, Dcs cs): Konstruktor
- Ugyfel getugyf()const: Visszaadja az ugyf nevû változót
- Dijcsomag getdijcs() const: Visszaadja az dijcs nevû változót
- Dcs getcsomag() const: Visszaadja az csomag nevû változót
- Rekord& operator=(const Rekord& r): = operátor, Rekord típusú adat elemeit másolja bele az osztály változóiba
- ~Rekord(): Destruktor

## Rekorlist osztály:

adattagok:

- static const size_t maxmeret = 100: tömb maximális mérete
- Rekord* lista[maxmeret]: Rekord* típusú maxméret nagyságú tároló tömb
- size_t db: A tömbben lévõ rekordok száma

tagfüggvények:

- RekordList(const RekordList& rhs): "Elrejtett" másoló konstruktor
- RekordList& operator=(const RekordList& rhs): Értékadó operátor
- RekordList(): Konstruktor
- size_t size() const: Visszaadja a db nevû változót
- void hozzaad(Rekord *r): Egy paraméterben megadott rekord pointert ad hozzá a listához a függvény
- void kilistaz() const: A listában lévõ összes rekordot kiírja a kimenetre
- void torol(int idx): Egy maraméterben megadott int típusú index megadásával kitörli az adott elemet a listából

- void modosit(int idx): Egy maraméterben megadott int típusú index megadásával a függvény lehetõvé teszi a felhasználó számára, hogy az adott indexû rekord valamelyik adattagját megváltoztassa
- Rekord& operator[](int idx): [] operátor, Rekordlista típusú változók indexelélét teszi lehetõvé
- ~RekordList(): Destruktor

## További függvények:

Fájlkezelő függvények:

- void fajl_beolvas(RekordList& L): A paraméterében megadott Rekodlista& típusú változóba beolvassa a két fülön fájlba megadott adatokat(ugyfel.txt, dijcsomag.txt)
- void fajlba_kiir(RekordList& L): A paraméterében megadott Rekodlistac típusú változóból írja ki az adatokat két külön fájlba (ugyfel.txt, dijcsomag.txt)

Menü kezelő függvények:

- void fomenu(RekordList& Lista): Kiírja a fõmenüt a kimenetre és a felhasználó elé tárja a választási lehetõségeket. A paraméterében megadott Rekodlista& típusú változót megadva használja a Rekodlista osztály további függvényeit
- void ugyfel_felvetel(RekordList& L): Uj rekordok felvételét teszi lehetõvé.A paraméterében megadott Rekodlista& típusú változóhoz adja hozzá a új rekordokat

Egyéb függvények:

- std::ostream& operator<<(std::ostream& os, const Cim& c): << operátor, kiírja a címhez tarozó összes adattagot
- bool operator==(const String& lhs, const String& rhs): Egyenlõség vizsgálat
- std::ostream& operator<<(std::ostream& os, const String& s): << operator, amivel a szabvanyos kimenetre irjuk a string osztály változóit
- std::istream& operator>>(std::istream& is, String& s): << operátor, ami beolvas az istreamről egy String típusú változót
- std::ostream& operator<<(std::ostream& os, const Dijcsomag& d): << operátor, kiírja a címhez tarozó összes adattagot
- std::ostream& operator<<(std::ostream& os, const Ugyfel& u): << operator, amivel a szabvanyos kimenetre irjuk a Ugyfel osztály változóit
- std::ostream& operator<<(std::ostream& os, const Rekord& c): << operátor, kiírja a rekordhoz tarozó összes adattagot

# A program felépítése:

Forrásfájlok:

main.cpp: A KAPCSOLO makro segítségével lehetkontrollálni, hogy a tesztesetek fussanal le, vagy pedig a felhasználói felület ugorjon fel
cim.h: Cim osztály definíciója, illetve az osztályhoz tartozó << operátor deklarációja található itt
cim.cpp: Cim osztály tagfüggvényeinek a definíciója, illetve az << operátor definíciója található itt
dijcsomag.h: A Dijcsomag főosztály és leszámazottainak, az Alap, MobilNet és SMSMax osztályoknak található itt a definíciója, valamint a Dijcsomag osztályhoz tartozó << operátor deklarációja vannak itt
dijcsomag.cpp: A főosztály és leszármazottjai tagfüggvényeinek definíciói illetve az ehhez tartozó << operátornak a definiciója található itt
fajlkezeles.h: A fájlkezeléhez használt két függvény deklarációja van ebben
fajlkezeles.cpp:A fájlkezelő függvények definíciója található itt
menu.h: A menu kiíratásához és működtetéséhez használt függvények deklarációja található itt
menu.cpp: A menu kiíratásához és működtetéséhez használt függvények definíciója található itt
rekord.h: A Rekord osztály definíciója és a hozzá tartozó << operátor deklarációja található itt
rekord.cpp: A Rekord osztály tagfüggvényeinek definíciója és a hozzá tartozó << operátor definíciója található itt
rekordlist.h: A Rekordlist osztály definíciója található itt
rekordlist.cpp: A Rekordlist osztály tagfüggvényeinek definíciója található itt
string.h: A String osztály definíciója, a hozzá tartozó << operátor, == operátor és >> operátor deklarációja található itt
string.cpp: A String osztály tagfüggvényeinek definíciója, a hozzá tartozó << operátor, == operátor és >> operátor definíciója található itt
ugyfel.h: A Ugyfel osztály definíciója és a hozzá tartozó << operátor deklarációja található itt
ugyfel.cpp: A Ugyfel osztály tagfüggvényeinek definíciója és a hozzá tartozó << operátor definíciója található itt

Nem saját fáljok:

gtest_lite.h, memtrace.h, memtrace.cpp

Adatfájlok:

dijcsomag.txt: Az ügyfelek dijcsomagainak az értékei vannaik itt eltárolva(Lebeszélt percek, küldött sms-ek)
ugyfel.txt: Az ügyfelek személyes adatai vannek ebben a fájlban eltárolva(név, város, utca, házszám, irányító szám, telefonszám, díjcosmagjának a típusa)