Tantárgy neve, kódja: Programozás II.; GINFBAN-PROGRAM2-1

Kapott program témája, sorszáma, címe:

02. Feladat: Egy futár heti szállításainak feldolgozása

Hallgató neve, Neptun kódja: *Csernák Gergely, TVJ2T2*
E-mail: *cserngergely01@gmail.com*

2020/2021. év II. félév
Neumann János Egyetem Műszaki és Informatikai Kar
2021.

Tartalom

BEVEZETÉS

A Programozás II. tantárgy hallgató kötelezettségének – zárthelyi dolgozat követelmények - teljesítése érdekében a tantárgy gyakorlati oktatója, Agg Péter által a 2. feladatsorszámú feladat megoldásának kitűzésére került sor 2021. április 19-én. A feladathoz kapcsolódóan rendelkezésre bocsátott feladat leírása szerint:

„A nagyvárosokon belül, ha csomagot gyorsan kell eljuttatni egyik helyről a másikra, akkor sokszor a legjobb választás egy kerékpáros futárszolgálat igénybevétele. A futárszolgálat a futárjainak a megtett utak alapján ad fizetést. Az egyik futár egy héten át feljegyezte fuvarjai legfontosabb adatait, és azokat eltárolta egy állományban. Az állományban az adatok rögzítése nem mindig követi az időrendi sorrendet. Azokra a napokra, amikor nem dolgozott, nincsenek adatok bejegyezve az állományba.”[1]

„A fájlban legalább 10 sor van, és minden sor egy-egy út adatait tartalmazza egymástól szóközzel elválasztva. Az első adat a nap sorszáma, ami 1 és 7 közötti érték lehet. A második szám a napon belüli fuvarszám, ami 1 és 40 közötti érték lehet. Ez minden nap l-től kezdődik, és az aznapi utolsó fuvarig egyesével növekszik. A harmadik szám az adott fuvar során megtett utat jelenti kilométerben, egészre kerekítve. Ez az érték nem lehet 30-nál nagyobb.”[2]

***„Készítsen programot**, amely a **tavok.txt** állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse **futar** néven! (A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.)”*[3]

A feladat teljesítéséhez szükséges fejlesztési és felhasználói dokumentációt - az e tárgyban kiadott Zárthelyi dolgozat követelmények[4] figyelembevételével - jelen irat egy-egy fejezete, a forráskódot pedig a melléklet tartalmazza.

[1] 2_feladat.pdf 1.oldal második bekezdése
[2] 2_feladat.pdf 1.oldal harmadik bekezdése
[3] 2_feladat.pdf 1.oldal ötödik bekezdése
[4] Beadando_kovetelmenyek_2021.pdf

FELHASZNÁLÓI DOKUMENTÁCIÓ

1. Program célja

1.1 Futár, mint foglalkozás ismertetése

Az emberiség története során a különféle anyagok mozgatása, kereskedelmi áruk szállítása mindig is fontos szerepet játszott, hatalmas háborúk dúltak fontos kereskedelmi útvonalakért és országok komoly gazdasági fellendülését vagy éppen összeomlását is jelenthette egy-egy ilyen útvonal birtoklása. Szerencsére a mai hétköznapi embernek már nincs oka ilyen miatt aggódni, hisz megszokottá vált, hogy ha szeretnénk valamit és azt meg is rendeljük az interneten, pár napon belül már az ajtónk előtt heverhet is.

Ahhoz, hogy ez a gyors és igen összetett logisztikai folyamat létrejöhessen, rengeteg ember munkája szükséges, kezdve attól, aki a csomagot összekészíti, egészen addig a futárig, aki becsönget egy kellemes kedd délután, hogy megérkezett a csomagunk.

A nagyvárosokban tekintettel az olykor borzalmas és hosszadalmas közlekedési dugókra, sokkal célszerűbb kerékpáros futárokat alkalmazni, mivel náluk nem áll fenn ennek a veszélye és a kisebb csomagokat így sokkal gyorsabban és „zöldebben” szállíthatnak ki a fogyasztóknak.

1.2. ’Futar’ program konkrét célja

A futárok munkájának nagyon fontos része az **adminisztráció**, ugyanis sok cég a megtett kilométer alapján ítéli meg az adott személy fizetését vagy nagyban befolyásolja azt. Természetesen a visszakövethetőség miatt is fontos, ugyanis sokszor nagy értékű termékeket is szükséges szállítani (ékszer, elektronikai eszközök stb.).

A **tavok.txt** tartalmazza egy futár munkájának egy heti munkáját feljegyezve, a **hét napjainak a sorszáma** (1-7), az adott napon **hányadik fuvarról beszélünk** (0-40) és a **csomag kézbesítéséért megtett távolságot** (km-ben).

Ezen adatok kezelése céljából merült fel az igény egy program megalkotására, az alábbi részcélokkal:

A rendelkezésre álló adatok felhasználásával a program jelenítse meg a képernyőn:

- A megtett távolságot
  - A hét első szállításánál (nem biztos, hogy a hét első napján volt)
  - A hét utolsó szállításánál
- A nap sorszámát
  - Amelyeken nem dolgozott (a fuvarszám 0)
  - Amelyiken a legtöbb fuvar teljesítette a futár
- Több információ
  - Naponként bontva a megtett távolság
  - Egy bekért távolság megállapítása, hogy mennyi pénzt keresne vele a futár
  - Kiírás a dijazas.txt fájlba, amelyben szükséges feltüntetni a nap és fuvarszámot, valamint a megkeresett összeget
  - Az egész heti fizetés összegzése

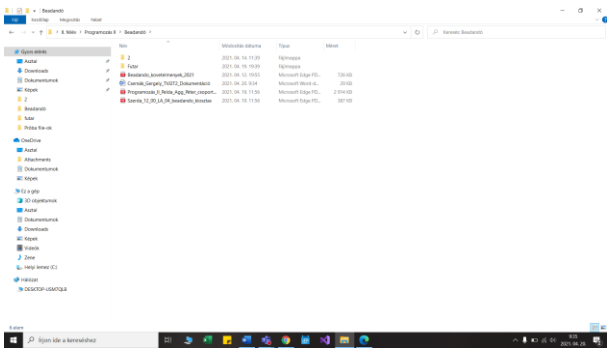
2. Program kezelés

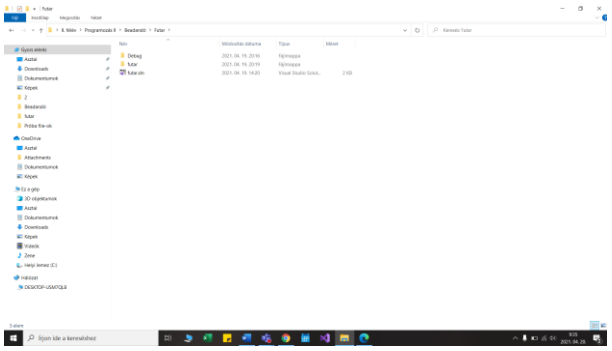
2.1 A program hardver és szoftver környezete

Hardver szükséglet szempontjából a **minimális memóriára, processzorra**, szabad lemezterületre van szükség. Képernyő felbontásnál **nem szükséges a 1024x768** felbontás, kisebb is elég. Külön lemezmeghajtót, külső mutató eszközt, kamerát, mikrofont nem igényel, viszont az asztali számítógép esetén a **billentyűzet szükséges**. A kiszolgáló szoftver környezet lényegében bármilyen exe operációs rendszer futtatására alkalmas operációs rendszer, így például Windows rendszerek is használhatóak, különösen a **Windows 10**. verzió. A program cpp kiterjesztési formája végett, a **Visual Studio 2019**-es alkalmazással történő fájl kezelését, indítását javasolnám.

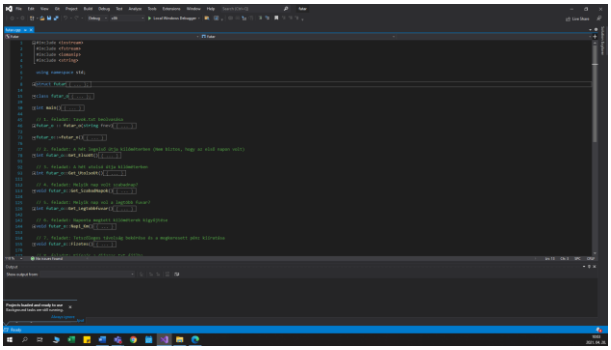
2.2 A program indítása

A program adatainak kezeléséhez szükséges a **futar.cpp és tavok.txt** fájlok átmásolása a **Visual Studio** által létrehozott új projekt – Futar -- mappába.





Majd ezt követően a Futar mappában lévő exe fájl megnyitásával a **Visual Studio** alkalmazásba jutunk, ahol a **futar.cpp** fájl indításával a programunk elindul.

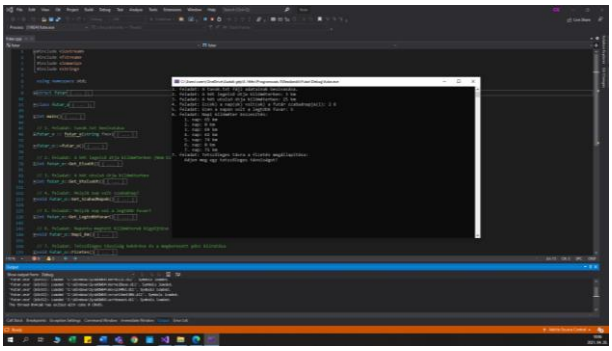


2.3 A program bemenete

S itt már meg is jelennek a képernyőn a rendelkezésre álló adatbázisból – jelen esetben – tavok.txt fájlból generált szűrési adatok:

- visszaigazolás a tavok.txt fájl beolvasásáról
- a hét legelső és utolsó útjának a távolságai
- a futár szabadnapjainak sorszáma felsorolva
- a legmozgalmasabb nap sorszáma (legtöbb fuvar)

- napokra bontva a megtett távolság összesítése



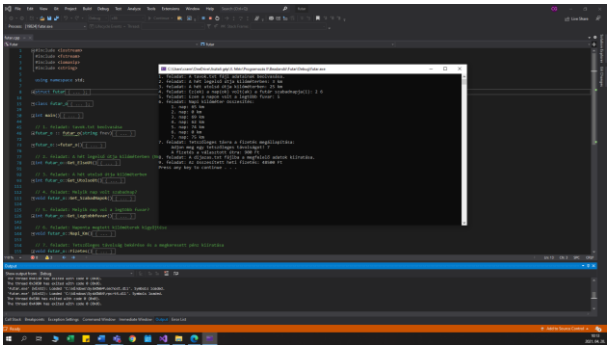
A következő lépésben egy általunk tetszőlegesen megadott távolság alapján megkaphatjuk azt az értéket, amennyi fizetést kap a futár egy ilyen távolságban lévő címért. Így alapvetően Program szempontjából két bemenetről beszélhetünk: a már meglévő adatok felhasználása és a lekérdezésnél történő távolság bevitele.

A folytatásban megtekinthetjük

- az előbbiekben kiválasztott távolság „értékét”
- visszaigazolás a dijazas.txt állomány létrehozásáról
- a heti fizetés összesítése

2.4 A program kimenete

A 2.3. pontban hivatkozott tetszőleges távolság megadása után a program folytatásában a következő végeredmény tekinthető meg:



Például: 7 km-es választlehetőségnél a „A fizetés a választott útra: 900 Ft” szöveg jelenik meg, amely 900 forintos fizetésre utal az általunk megadott útért.

Továbbá kimenetként: a dijazas.txt állományban létrehozza a futár heti munkájának az összefoglalóját:

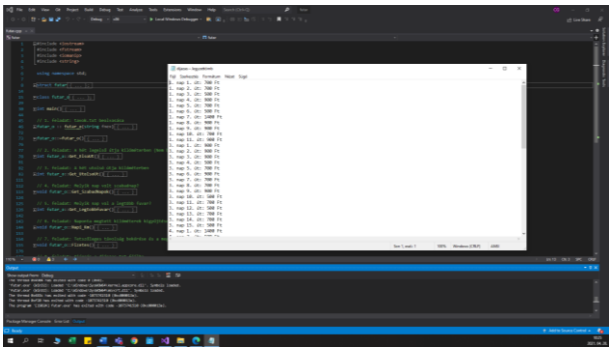
„…nap szerint, azon belül pedig az út sorszáma szerinti növekvő sorrendben az alábbi formátumban:

1. nap 1. út: 700 Ft

1. nap 2. út: 900 Ft

1. nap 3. út: 2000 Ft”[5]

A létrehozottdijazas.txt állományt a program a Futar mappában hozza létre. A dijazas.txt tartalma:



[5] 2_feladat.pdf 2. oldal 8. feladat

FEJLESZTŐI DOKUMENTÁCIÓ

1. A program által megvalósítani kívánt feladatok, kitűzött célok

A futár heti teljesítményét kezelő program a rendelkezésre álló adatok felhasználásával jelenítse meg a képernyőn:

- a hét legelső és utolsó útjának a távolságait
- a futár szabadnapjainak sorszámát felsorolva
- a legmozgalmasabb nap sorszámát (legtöbb fuvar)
- napokra bontva a megtett távolságot összesítve
- egy tetszőlegesen megadott távolság „értékét”

Az adatbázisban hozza létre egy ’dijazas’ végeredményt, ahol nap szerint, azon belül pedig az út sorszáma szerinti növekvő sorrendben tüntesse az adatokat az alábbi fomában:

„1. nap 1. út: 700 Ft

1. nap 2. út: 900 Ft

1. nap 3. út: 2000 Ft”[6]

2. Előzetes rendszerterv

2.1A feladathoz szükséges állapotfelmérés

Az adatbázis teljes adatállományának felmérése, azok típusonkénti rendszerezése. Ugyanis nem biztos, hogy a lekérdezések szerinti bontások szűrhetők, amennyiben ezt nem rendszerezzük.

[6] 2_feladat.pdf 2. oldal 8. feladat

2.2A feladat megoldáshoz szükséges erőforrás meghatározása

Becsléssel történő időigény meghatározása analógiával: más szoftver fejlesztések során alkalmazott ráfordított idő vagy szakértői vélemény, esetleg más modell alapú számításokkal. Amennyiben célszerű: Gantt diagram készítése.

2.3Követelmény menedzsment folyamata

Célja a projekt által előállított termékre vonatkozó követelmények elemzése, nyilvántartása es azok összeférhetetlenségének kiszűrésére. A nem megvalósítható követelményeket fel kell fedni, es a megrendelővel egyeztetni kell a további sorsukról. A követelmények megadásának es gyűjtésének formalizálása. Az egyes alrendszerek, modulok követelményeinek nyilvántartása a tervezés minden fázisában fontos. Sokszor szükség lehet arra, hogy fejlesztés közbeni követelményváltozásra reagáljunk.

Ezért fokozottan fontos, hogy nyomon kövessük, egyrészt a felhasználói követelményrendszer változását, másrészt azt, hogy a felhasználói követelmények közül melyik hova lett beépítve.

2.4. Alternatív megoldások vizsgálata

Legjobb megoldás kiválasztása, a kockázatok feltérképezése, megelőzése.

2.5. Választott program megvalósítása

Beleértve és figyelembe véve a megrendelői környezet adottságait, így a program nyelvének megfelelő megválasztását is.

3. Részletes rendszerterv

3.1. Képernyőtervek

3.1.1. Program bemeneti képernyőterv

A képernyő színmegjelenítése fekete-fehér lesz, viszont a képernyőterv szemléltetése miatt célszerűnek látszik ezek eltérő ábrázolása.

a) Az első képernyőterv a rendelkezésre álló adatbázisból – jelen esetben –tavok.txt fájlból generált szűrési adatok megjelenítése:

A visszaigazolás a tavok.txt állomány beolvasásáról (barna színnel) [A tavok.txt fájl adatainak beolvasása.]

A hét legelső útja kilométerben (barna színnel),

A hét utolsó útja kilométerben (barna színnel),

Ez(ek) a nap(ok) volt(ak) a futár szabadnapja(i) (barna színnel),

Ezen a napon volt a legtöbb fuvar (barna színnel),

Napi kilométer összesítés (barna színnel)

és az ezekre válaszként érkező értékek (zöld színnel).

A hét legelső útja kilométerben:	értékek kiírása
A hét utolsó útja kilométerben:	értékek kiírása
Ez(ek) a nap(ok) volt(ak) a futár szabadnapja(i):	értékek kiírása (1) , (2)
Ezen a napon volt a legtöbb fuvar:	értékek kiírása
Napi kilométer összesítés:	értékek kiírása (1)
	értékek kiírása (2)
	értékek kiírása (3)
	értékek kiírása (4)
	értékek kiírása (5)
	értékek kiírása (6)
	értékek kiírása (7)

Első képernyőterv

b) A második képernyőterv: A következő lépésben egy szám bekérése következik, egy tetszőleges távolságot szükséges megadni a felhasználónak. A billentyűzettel történő bekérést (piros színnel) ábrázolom.

A hét legelső útja kilométerben:	értékek kiírása
A hét utolsó útja kilométerben:	értékek kiírása

Ez(ek) a nap(ok) volt(ak) a futár szabadnapja(i):	értékek kiírása (1) , (2)
Ezen a napon volt a legtöbb fuvar:	értékek kiírása
Napi kilométer összesítés:	értékek kiírása (1)
	értékek kiírása (2)
	értékek kiírása (3)
	értékek kiírása (4)
	értékek kiírása (5)
	értékek kiírása (6)
	értékek kiírása (7)
Tetszőleges távra a fizetés megállapítása	
Adjon meg egy tetszőleges távolságot!	érték begépelése

Második képernyőterv

c) A harmadik képernyőterv: a tetszőlegesen megadott távolság méretétől függően a program megvalósítja és megadja azt az összeget, amely az adott távolsághoz tartozik a bérezés területén. Ezt (világoskék színnel) jelölöm.

A hét legelső útja kilométerben:	értékek kiírása
A hét utolsó útja kilométerben:	értékek kiírása
Ez(ek) a nap(ok) volt(ak) a futár szabadnapja(i):	értékek kiírása (1) , (2)
Ezen a napon volt a legtöbb fuvar:	értékek kiírása
Napi kilométer összesítés:	értékek kiírása (1)
	értékek kiírása (2)
	értékek kiírása (3)
	értékek kiírása (4)
	értékek kiírása (5)
	értékek kiírása (6)
	értékek kiírása (7)
Tetszőleges távra a fizetés megállapítása	
Adjon meg egy tetszőleges távolságot!	érték begépelése

	A fizetés a választott útra:	értékek kiírása	

Harmadik képernyőterv

d) A negyedik képernyőterv: ezután még megjelenik egy visszaigazoló üzenet a dijazas.txt létrejöttéről, valamint a futár fizetésének egész heti összegzése (barna színnel).

A hét legelső útja kilométerben:	értékek kiírása
A hét utolsó útja kilométerben:	értékek kiírása
Ez(ek) a nap(ok) volt(ak) a futár szabadnapja(i):	értékek kiírása (1) , (2)
Ezen a napon volt a legtöbb fuvar:	értékek kiírása
Napi kilométer összesítés:	értékek kiírása (1)
	értékek kiírása (2)
	értékek kiírása (3)
	értékek kiírása (4)
	értékek kiírása (5)
	értékek kiírása (6)
	értékek kiírása (7)
Tetszőleges távra a fizetés megállapítása	
Adjon meg egy tetszőleges távolságot!	érték begépelése
A fizetés a választott útra:	értékek kiírása
A dijazas.txt fájlba a megfelelő adatok kiíratása.	
Az összesített heti fizetés:	értékek kiírása

Negyedik képernyőterv

f) További kimenetként: a dijazas.txt állományban létrehozza a futár heti bérezésének az összefoglalóját, ahol soronként feltünteti a futár minden egyes fuvarjának a bérezését a nap majd a fuvarszám szerint sorba rendezve. Ez ugyan nem jelenik képernyőképként, de jegyzettömbként történő megjelenítésének terve a következő:

A nap sorszáma		A fuvar sorszáma		A megkeresett összeg	
1.	„nap”	1.	„út”	700	„Ft”
1.	„nap”	2.	„út”	700	„Ft”
…	…	…	…	…	…
7.	„nap”	8.	„út”	2000	„Ft”

3.2. Felhasznált változók felírása

	Forráskódban előforduló változók	Típusa	Funkciója, használata
1	int nap_sorszama	Alaptípus – Skalár (előjeles 32 bites egész)	a hét napjai sorszámozva (1-7)
2	int fuvarszam		egy adott napon az adott fuvar sorszámát adja meg (1-40)
3	int km		egy adott fuvarszámhoz tartozó távolság (km-ben)
4	int penz		egy adott fuvarhoz tartozó fizetség
5	int db		a tavok.txt fájlban lévő infomáció egységek darabszáma
5	int i = 0		léptetéshez (számláláshoz) szükséges - egész szám használata
6	int j = 0		léptetéshez (számláláshoz) szükséges - egész szám használata
8	int ElsoKm		a hét legelső útját foglalja magába (km)
9	int ElsoNap		a hét első munkanapjának a sorszámát írja le
10	int UNap		a hét utolsó munkanapjának a sorszáma
11	int UFuvar		a hét utolsó fuvarjának a sorszáma
12	int UKm		a hét utolsó fuvarján megtett távolság (km)
13	int Napi_Fuvar		egy tömb, amely a napi fuvarok számát tartalmazza
14	int Max_Fuvar		az egy nap alatt teljesített legtöbb fuvar száma
15	int Napi_km		tömb, amely napokra bontva a megtett távolságot tartalmazza
16	int T_km		a felhasználó által megadott tetszőleges távolág

17	int fizetes		a tetszőlegesen megadott távolsághoz tartozó fizetés (Ft)
18	int Heti_fizu		az egész heti munkáért összesített fizetés összege

3.3. Algoritmus-függvény bemutatására

Kiválasztó algoritmus

Ilyen jellegű algoritmus a beolvasás része a forráskódnak a tavok.txt fájlból: „Olvassa be a tavak.txt állományban talált adatokat, s annak felhasználásával oldja meg a következő feladatokat!” Azonban előtte osztály létrehozása, indítása: a privátba rakom, amit csak az osztály használhat; a publikusba pedig a többit:

```
class futar_o {
private: futar* A;
        int db;
public: futar_o(string fnev);
       ~futar_o();
       int Get_ElsoUt();
       int Get_UtolsoUt();
       void Get_SzabadNapok();
       int Get_Legtobbfuvar();
       void Napi_Km();
       void Fizetes();
       int Heti_Fizetes();
       void Kiir();
};
```

Cserélő-rendező algoritmus:

S maga a verseny.txt fájlból történő beolvasás, valamint utána az adatok sorba rendezése a nap és a fuvarszám szerint:

```
ifstream be(fnev);
   if (be.fail()) { cerr << "Hiba a fájl beolvasása közben!"; system("pause"); exit(1); }

   A = new futar[100];
   for (int i = 0; !be.eof(); i++)
   {
      be >> A[i].nap_sorszama;
      be >> A[i].fuvarszam;
      be >> A[i].km;
      db++;
      // cout << setw(3) << A[i].nap_sorszama << setw(3) << A[i].fuvarszam << setw(3) <<
A[i].km << endl; // Önellenőrzés, a beolvasott fájlok megtekintésére
```

```
    }
    be.close();

    // Rendezés
    for (int i = 0; i < db - 1; i++) {
        for (int j = i + 1; j < db; j++) {
            futar csere;
            if (A[i].nap_sorszama > A[j].nap_sorszama || (A[i].nap_sorszama ==
A[j].nap_sorszama && A[i].fuvarszam > A[j].fuvarszam)) {
                csere = A[i];
                A[i] = A[j];
                A[j] = csere;
            }
        }
    }
```

A már említett osztály lezárását követően kerül sor a destruktor függvény futatására:

```
futar_o::~futar_o() {
    if (A != 0) { delete[]A; }
}
```

Maximum kereső algoritmus

Ilyen algoritmus található az 5. feladatnál, ahol annak a napnak a sorszámát kell megadnunk, amelyiken a legtöbb fuvart végezte a futár.

```
// 5. feladat: Melyik nap volt a legtöbb fuvar?
int futar_o::Get_Legtobbfuvar() {
    int Max_Fuvar = 1;
    int Napi_Fuvar[8] = { 0 };
    for (int i = 0; i < db; i++)
    {
        Napi_Fuvar[A[i].nap_sorszama]++;
    }
    for (int i = 2; i <= 7; i++)
    {
        if (Napi_Fuvar[i] > Napi_Fuvar[Max_Fuvar])
        {
            Max_Fuvar = i;
        }
    }
    return Max_Fuvar;
}
```

3.4. Tesztelés

A programot írás közben is, részfeladatonként ellenőriztem, teszteltem. A program írása folyamán olykor ellenőrzés gyanánt alkalmaztam kiíratásokat a képernyőre, amelyeket a gondolatmenetem helyességének igazolása ként írtam bele a kódba, amennyiben az elvárásnak megfelelő volt az eredmény, ezeket kommentbe helyeztem vagy töröltem a programból. Például a beolvasás után a fájlok kiíratása (db++ sor alatt):

```
// 1. feladat: tavok.txt beolvasása
futar_o :: futar_o(string fnev) {
    ifstream be(fnev);
    if (be.fail()) { cerr << "Hiba a fájl beolvasása közben!"; system("pause"); exit(1); }
    A = new futar[100];
    for (int i = 0; !be.eof(); i++)
    {
        be >> A[i].nap_sorszama;
        be >> A[i].fuvarszam;
        be >> A[i].km;
        db++;
        // cout << setw(3) << A[i].nap_sorszama << setw(3) << A[i].fuvarszam << setw(3) << A[i].km << endl; // Önellenőrzés, a beolvasott fájlok megtekintésére
    }
    be.close();
```

A program futtatása során szinte mindegyik feladat a tavok.txt fájlból beolvasott adatokkal dolgozik, kivéve a 7. feladat során, amelynél a tetszőlegesen megadott távolságtól függően alakul a feladat folytatása. A tesztelés során az 5-ös értéket adtam meg, amelyre a program az elvárt választ, 700-at hozott eredményül.

MELLÉKLET: A TELJES FORRÁSKÓD

```
#include <iostream>
#include <fstream>
#include <iomanip>

using namespace std;

struct futar {
    int nap_sorszama; // A nap sorszáma, ami 1 és 7 közötti érték lehet
    int fuvarszam; //  A napon belüli fuvarszám, ami 1 és 40 közötti érték lehet
    int km; // Az adott fuvar során megtett utat jelenti kilométerben
    int penz; // Később bevezetett érték
};

class futar_o {
private: futar* A;
         int db;
public: futar_o(string fnev);
        ~futar_o();
        int Get_ElsoUt();
        int Get_UtolsoUt();
        void Get_SzabadNapok();
        int Get_Legtobbfuvar();
        void Napi_Km();
        void Fizetes(int T_km);
        int Heti_Fizetes(int Heti_fizu);
        void Kiir();
};

int main() {
    setlocale(LC_ALL, "");
    int T_km;
    int Heti_fizu = 0;
    cout << "1. feladat: A tavok.txt fájl adatainak beolvasása. " << endl; futar_o
adatok("tavok.txt");
    cout << "2. feladat: A hét legelső útja kilométerben: " << adatok.Get_ElsoUt() << " km"
<<endl;
    cout << "3. feladat: A hét utolsó útja kilométerben: " << adatok.Get_UtolsoUt() << "
km" << endl;
    cout << "4. feladat: Ez(ek) a nap(ok) volt(ak) a futár szabadnapja(i): " ;
adatok.Get_SzabadNapok(); cout << endl;
    cout << "5. feladat: Ezen a napon volt a legtöbb fuvar: " << adatok.Get_Legtobbfuvar()
<< endl;
    cout << "6. feladat: Napi kilométer összesítés: " << endl; adatok.Napi_Km();
    cout << "7. feladat: Tetszőleges távra a fizetés megállapítása: " << endl;
    cout << "     " << "Adjon meg egy tetszőleges távolságot!  " ; cin >> T_km;
adatok.Fizetes(T_km);
    cout << "8. feladat: A dijazas.txt fájlba a megfelelő adatok kiíratása. " << endl;
adatok.Kiir();
    cout << "9. feladat: Az összesített heti fizetés: " << adatok.Heti_Fizetes(Heti_fizu)
<< " Ft" << endl;
    system("pause");
    return 0;
}

// 1. feladat: tavok.txt beolvasása
futar_o :: futar_o(string fnev) {
    ifstream be(fnev);
    if (be.fail()) { cerr << "Hiba a fájl beolvasása közben!"; system("pause"); exit(1); }
    A = new futar[100];
    for (int i = 0; !be.eof(); i++)
```

```
    {
        be >> A[i].nap_sorszama;
        be >> A[i].fuvarszam;
        be >> A[i].km;
        db++;
        // cout << setw(3) << A[i].nap_sorszama << setw(3) << A[i].fuvarszam << setw(3) <<
A[i].km << endl; // Önellenőrzés, a beolvasott fájlok megtekintésére
    }
    be.close();

    // Rendezés
    for (int i = 0; i < db - 1; i++) {
        for (int j = i + 1; j < db; j++) {
            futar csere;
            if (A[i].nap_sorszama > A[j].nap_sorszama || (A[i].nap_sorszama ==
A[j].nap_sorszama && A[i].fuvarszam > A[j].fuvarszam)) {
                csere = A[i];
                A[i] = A[j];
                A[j] = csere;
            }
        }
    }
}

futar_o::~futar_o() {
    if (A != 0) { delete[]A; }
}

// 2. feladat: A hét legelső útja kilóméterben (Nem biztos, hogy az első napon volt)
int futar_o::Get_ElsoUt() {
    int ElsoKm = 0;
    int ElsoNap = 7;
    for (int i = 0; i < db; i++)
    {
        if (A[i].nap_sorszama < ElsoNap) { ElsoNap = A[i].nap_sorszama; } //
Minimumkiválasztással kiderítjük, hogy melyik nap volt az első munkanapja a futárunknak
    }
    for (int i = 0; i < db; i++)
    {
        if (ElsoNap == A[i].nap_sorszama && A[i].fuvarszam == 1) { ElsoKm = A[i].km; }
    }
    return ElsoKm;
}

// 3. feladat: A hét utolsó útja kilóméterben
int futar_o::Get_UtolsoUt() {
    int UNap = 1;
    int UFuvar = 1;
    int UKm = 0;
    for (int i = 0; i < db; i++)
    {
        if (A[i].nap_sorszama > UNap) { UNap = A[i].nap_sorszama; } //
Maximumkiválasztással kiderítjük, hogy melyik nap volt az utolsó munkanapja a futárunknak
    }
    for (int i = 0; i < db; i++)
    {
        if (A[i].nap_sorszama == UNap) {
            for (int j = 0; j < 40; j++) {
                if (A[i].fuvarszam > UFuvar) { UKm = A[i].km; } // Maximumkiválasztással
kiderítjük, hogy melyik volt az utolsó fuvarja a futárunknak
            }
        }
    }
    return UKm;
}
```

```
// 4. feladat: Melyik nap volt szabadnap?
void futar_o::Get_SzabadNapok() {
    int Napi_Fuvar[8] = { 0 };
    for (int i = 0; i < db; i++)
    {
        Napi_Fuvar[A[i].nap_sorszama]++;
    }
    for (int i = 1; i <= 7; i++)
    {
        if (Napi_Fuvar[i] == 0) { cout << i << " "; }
    }
}

// 5. feladat: Melyik nap volt a legtöbb fuvar?
int futar_o::Get_Legtobbfuvar() {
    int Max_Fuvar = 1;
    int Napi_Fuvar[8] = { 0 };
    for (int i = 0; i < db; i++)
    {
        Napi_Fuvar[A[i].nap_sorszama]++;
    }
    for (int i = 2; i <= 7; i++)
    {
        if (Napi_Fuvar[i] > Napi_Fuvar[Max_Fuvar])
        {
            Max_Fuvar = i;
        }
    }
    return Max_Fuvar;
}

// 6. feladat: Naponta megtett kilóméterek kigyűjtése
void futar_o::Napi_Km() {
    int Napi_km[8] = {0};
    for (int i = 0; i < db; i++) {
        Napi_km[A[i].nap_sorszama] += A[i].km;
    }
    for (int i = 1; i <= 7; i++) {
        cout << "     " << i << ". nap: " << Napi_km[i] << " km" << endl;
    }
}

// 7. feladat: Tetszőleges távolság bekérése és a megkeresett pénz kiíratása
void futar_o::Fizetes(int T_km) {
    int fizetes;
    if (T_km <= 2) {
        fizetes = 500;
    }
    else if (T_km <= 5) {
        fizetes = 700;
    }
    else if (T_km <= 10) {
        fizetes = 900;
    }
    else if (T_km <= 20) {
        fizetes = 1400;
    }
    else if (T_km <= 30) {
        fizetes = 2000;
    }
    cout << "     " << "A fizetés a választott útra: " << fizetes << " Ft" << endl;
}

// 8. feladat: Kiíraás a dijazas.txt fájlba
void futar_o::Kiir() {
    for (int i = 0; i < db; i++)
```

```
    {
        if (A[i].km <= 2) {
            A[i].penz = 500;
        }
        else if (A[i].km <= 5) {
            A[i].penz = 700;
        }
        else if (A[i].km <= 10) {
            A[i].penz = 900;
        }
        else if (A[i].km <= 20) {
            A[i].penz = 1400;
        }
        else if (A[i].km <= 30) {
            A[i].penz = 2000;
        }
    }
    ofstream ki("dijazas.txt");
    for (int i = 0; i < db; i++) {
        ki << A[i].nap_sorszama << ". nap " << A[i].fuvarszam << ". út: " << A[i].penz << "
Ft" << endl;
    }
    ki.close();
}

// 9. feladat: Futárunk fizetése a heti munkájáért
int futar_o::Heti_Fizetes(int Heti_fizu) {
    for (int i = 0; i < db; i++) {
        Heti_fizu += A[i].penz;
    }
    return Heti_fizu;
}
```

IRODALOMJEGYZÉK

1. **Agg Péter (2021)** *Programozás_II_Pelda_Agg_Péter_csoportja.pdf* Kecskemét: Neumann János Egyetem GAMF Műszaki és Informatikai Kar, Informatikai Tanszék.

2. **Dr. Pásztor Attila (2021)** *Programozás_II_3_4_előadás_Pásztor.pptm* Kecskemét: Neumann János Egyetem GAMF Műszaki és Informatikai Kar, Informatikai Tanszék.

3. **Agg Péter (2021)** *Beadando_kovetelmenyek_2021.pdf* Kecskemét: Neumann János Egyetem GAMF Műszaki és Informatikai Kar, Informatikai Tanszék.

4. **Agg Péter (2021)** *2_feladat.pdf* Kecskemét: Neumann János Egyetem GAMF Műszaki és Informatikai Kar, Informatikai Tanszék.