**Tartalom**

[Fejlesztői dokumentáció 1](#_heading=h.gjdgxs)

[Feladat leírása: 1](#_heading=h.30j0zll)

[Rendszerterv 2](#_heading=h.1fob9te)

[utak.txt állomány szerkezete 2](#_heading=h.3znysh7)

[Képernyőterv 3](#_heading=h.2et92p0)

[Fuvar struktúratípus: 3](#_heading=h.tyjcwt)

[Feldolgoz objektumtípus: 4](#_heading=h.3dy6vkm)

[Változótábla 4](#_heading=h.1t3h5sf)

[A felhasznált függvények leírása 5](#_heading=h.4d34og8)

[int main() 5](#_heading=h.2s8eyo1)

[Feldolgoz 5](#_heading=h.17dp8vu)

[1. feladat: void beolvas() 6](#_heading=h.3rdcrjn)

[2. feladat: int hetMasodikFuvarKm() 6](#_heading=h.26in1rg)

[3. feladat: int hetUtolsoFuvarKm() 6](#_heading=h.lnxbz9)

[4. feladat: void hetSzabadNapjai() 6](#_heading=h.35nkun2)

[5. feladat: int legtobbFuvarNap() 7](#_heading=h.1ksv4uv)

[6. feladat: void osszegezNaponkent() 7](#_heading=h.t9zemc33p26p)

[void szamolNapokat() 7](#_heading=h.44sinio)

[int szamolOsszesenKiirReszosszeg() 7](#_heading=h.2jxsxqh)

[int szamolTavAlapjan(int tav) 8](#_heading=h.z337ya)

[Felhasználói dokumentáció 8](#_heading=h.3j2qqm3)

[1. Hardver és szoftver követelmények: 8](#_heading=h.1y810tw)

[2. Program kezelése 9](#_heading=h.4i7ojhp)

[3. Tesztelés 10](#_heading=h.2xcytpi)

[4. Továbbfejlesztési lehetőségek 11](#_heading=h.1ci93xb)

[Irodalomjegyzék 11](#_heading=h.3whwml4)

[Forráskód 11](#_heading=h.2bn6wsx)

# Fejlesztői dokumentáció

## Feladat leírása:

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.

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például: 3. feladat:)! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár! Az ékezetmentes kiírás is elfogadott.

**Feladatok:**

1. Olvassa be az **utak.txt** állományban talált adatokat, s annak felhasználásával oldja meg a következő feladatokat!

2. Írja ki a képernyőre, hogy mekkora volt a hét második útja kilométerben! Figyeljen arra, hogy olyan állomány esetén is helyes értéket adjon, amiben például a hét első napján a futár nem dolgozott!

3. Írja ki a képernyőre, hogy mekkora volt a hét utolsó útja kilométerben!

4. Tudjuk, hogy a futár minden héten tart legalább egy szabadnapot. Írja ki a képernyőre, hogy a hét hányadik napjain nem dolgozott a futár!

5. Írja ki a képernyőre, hogy a hét melyik napján volt a legtöbb fuvar! Amennyiben több nap is azonos, maximális számú fuvar volt, elegendő ezek egyikét kiírnia.

6. Számítsa ki és írja a képernyőre a mintának megfelelően, hogy az egyes napokon hány kilométert kellett tekerni!

1. nap: 124 km

2. nap: 0 km

3. nap: 75 km

7. A futár az egyes utakra az út hosszától függően kap fizetést az alábbi táblázatnak megfelelően:

1 – 2 km 500 Ft

3 – 5 km 700 Ft

6 – 10 km 900 Ft

11 – 20 km 1 400 Ft

21 – 30 km 2 000 Ft

Kérjen be a felhasználótól egy tetszőleges távolságot, és határozza meg, hogy mekkora díjazás jár érte! Ezt írja a képernyőre!

8. Határozza meg az összes rögzített út ellenértékét! Ezeket az értékeket írja ki a **befizetes.txt** állományba nap szerint, azon belül pedig az út sorszáma szerinti csökkenő sorrendben, az alábbi formátumban:

1. nap 1. út: 2000 Ft

1. nap 2. út: 900 Ft

1. nap 3. út: 700 Ft

9. Határozza meg, és írja ki a képernyőre, hogy a futár mekkora összeget kap a heti munkájáért!

# Rendszerterv

## utak.txt állomány szerkezete

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 1-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.

**Például:**

1 1 5

1 2 9

3 2 12

1 4 3

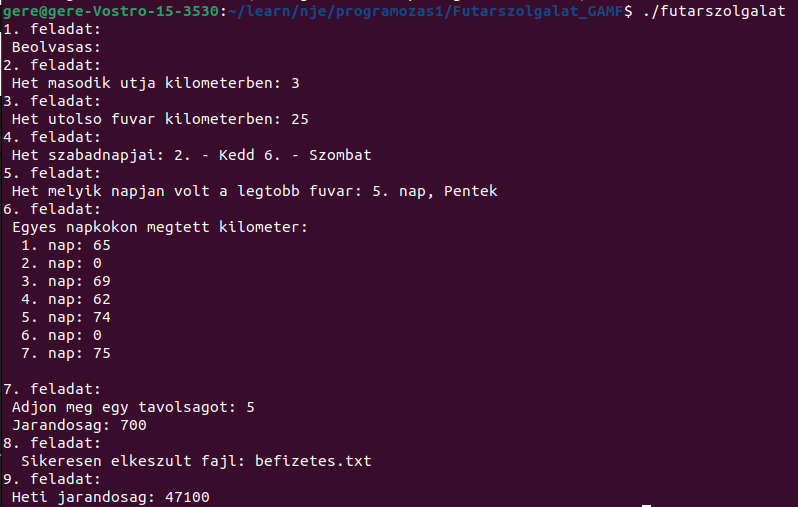
3 1 7

…

A 3. sor például azt mutatja, hogy a hét harmadik napján a második fuvar 12 kilométeres távolságot jelentett. Készítsen programot, amely az **utak.txt** állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse **futarszolgalat** néven! (A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.)

## Képernyőterv

Feladathoz nincs előírt grafikus felület, parancssoros felület képernyőképe Linuxon a következő:



## Fuvar struktúratípus:

Az utak.txt sorainak beolvasása a Fuvar nevű struktúrába történik a feldolgozása során.

Adattagjai:

| Név | Típus | leírás |
| --- | --- | --- |
| nap | int | Hét napjai számokkal kifejezve |
| sorszam | int | Napon belüli sorszám |
| tavolsag | int | A fuvar alatt megtett távolság |

## Feldolgoz objektumtípus:

A feldolgozás lépéseit implementáló objektum mely tagfüggvényei reprezentálják az elvárt működés feladatok lépéseit:

Tagfüggvények:

| Megnevezés | Leírás | Láthatóság |
| --- | --- | --- |
| Konstruktor | Létrehozáskori napok tömb init | public |
| beolvas | 1. feladat: beolvasás | public |
| hetMasodikFuvarKm | 2. feladat: hét második fuvar kilométer | public |
| hetUtolsoFuvarKm | 3. feladat: hét utolsó útja kilométerben | public |
| hetSzabadNapjai | 4. feladat: hét szabad napjai | public |
| legtobbFuvarNap | 5. feladat: hét melyik napján volt a legtöbb fuvar | public |
| osszegezNaponkent | 6. feladat: Naponkénti összegzés | public |
| szamolTavAlapjan | 7. feladat: Beadott távolság alapján fizetés | public |
| szamolOsszesenKiirReszosszeg | 8.-9.feladat: A futár mekkora összeget kap a heti munkájáért, utankénti összeget kiírja a befizetes.txt nevű fájlba | public |
| szamolNapokat | Napok szerinti felösszegzés | private |
| getNapStr | Nap meghatározása sorszám index alapján | public |
| calcErtek | Fuvar struct példány alapján kalkulja a rendezési értéket | private, static |
| rendez | Tömb elemeinek rendezése | private |

## Változótábla

| Név | Típus | Hatókör | leírás |
| --- | --- | --- | --- |
| fd | Feldolgoz | main | Feldolgoz objektum példány |
| tav | int | main | Távolságot változó bekéréshez |
| fuvarok | Fuvar tömb | Feldolgoz példány | Max 100 fuvar elemet tartalmazó tömb |
| db | int | Feldolgoz példány | Fájl beolvasáskor megszámoljuk a sorokat |
| napok | string tömb | Feldolgoz példány | Hét napjait táróló tömb 1-Hétfő |
| naponkentiDb | int tömb | Feldolgoz példány | Hét napontkénti fuvarok száma |
| naponkentiKm | int tömb | Feldolgoz példány | Hét napontkénti fuvarok össz. hossza |
| be | ifstream | beolvas | Ifstream példány |
| i | int | beolvas | Tömb index |
| ertek1 | int | hetMasodikFuvarKm | Min. kiválasztáshoz használt érték |
| ertek2 | int | hetMasodikFuvarKm | 2. Min. kiválasztáshoz használt érték |
| i1 | int | hetMasodikFuvarKm | Min. kiválasztáshoz használt index |
| i2 | int | hetMasodikFuvarKm | 2. Min. kiválasztáshoz használt index |
| maxNap | int | hetUtolsoFuvarKm | Utolsó nap választó változó |
| maxSorszamIndex | int | hetUtolsoFuvarKm | Utolsó fuvar választó változó, tömb index |
| n | int | hetSzabadNapjai | aktuális nap index |
| i | int | hetSzabadNapjai | aktuális fuvar index |
| maxIndex | int | legtobbFuvarNap | Legnagyobb távolság választó változó |
| n | int | osszegezNaponkent | aktuális nap index |
| fiz | int | szamolOsszesenKiirReszosszeg | Fizetség összegző változó |
| ki | ofstream | szamolOsszesenKiirReszosszeg | Ofstream példány |
| i | int | szamolOsszesenKiirReszosszeg | aktuális fuvar index |
| fiz | int | szamolTavAlapjan | Fizetés meghatározása változó |

## A felhasznált függvények leírása

### int main()

Belépő függvény amely kapcsolatot biztosít a terminál ablakból való hívást. Tartalmazza a feldolgozást végző függvényeket.

### Feldolgoz

Feldolgoz objektum konstruktor mely feladata, hogy feltöltse a nap string-eket az index alapján.

### 1. feladat: void beolvas()

Fájl utak.txt megnyitása és soronként végig olvasása a tartalmát, és feltölti a fuvarok tömbjét.

### 2. feladat: int hetMasodikFuvarKm()

Hét második fuvar távolságának meghatározása. Megoldás lehetett volna a rendezés után a második elem kiválasztása. De kétszer végrehajtva a minimum választást algoritmust választottam, annyi módosítással, hogy minimum választás első lépéseként használt minimum alapérték meghatározása helyett

### 3. feladat: int hetUtolsoFuvarKm()

Hét utolsó fuvar távolságának meghatározása, melyhez két maximum kiválasztást használunk:

Be:   
 fuvarok Fuvar tömb  
 db tömbben található elemek száma

Algoritmus:  
 Ciklus i legyen 0-tól míg kisebb mint db egyesével  
 Ha fuvarok[i].nap nagyobb maxNap akkor maxNap legyen fuvarok[i].nap Ciklus vége.   
 Ciklus i := 0-tól míg kisebb mint db egyesével  
 Ha fuvarok[i].nap egyenlő maxNap   
 és fuvarok[i].sorszam nagyobb mint a maxSorszamIndex-ő sorszama akkor maxSorszamIndex := i;  
 Ciklus vége.   
 Vissza: fuvarok[maxSorszamIndex].tavolsag;  
Algoritmus vége.

### 4. feladat: void hetSzabadNapjai()

Minden egyes napon végigmenve megvizsgáljuk eldöntés tétele, hogy az adott nap szabad nap volt-e:

Be:   
 fuvarok Fuvar tömb  
 db tömbben található elemek száma

Algoritmus:  
 Ciklus n legyen 1-től 7-ig egyesével  
 i:=0  
 Ciklus míg i < db és fuvarok[i].nap != n  
 i:= i +1  
 Ciklus vége  
 Ha i == db akkor kiír: “n.nap szabadnap”  
 Ciklus vége  
Algoritmus vége

### 5. feladat: int legtobbFuvarNap()

Részfeladatként elvégezzük a naponkénti távolságok felösszegzést lsd.: szamolNapokat fv. Majd egy maximum választással meghatározzuk a legtöbb számú fuvarral rendelkező napot.

Be:   
 naponkentiDb int tömb  
Algoritmus:  
 maxIndex := 1;   
 Ciklus i legyen 2-től 7-ig egyesével  
 Ha napponkentiDb[i] nagyobb mint napponkentiDb[maxIndex] akkor maxIndex:=i;  
 Ciklus vége.  
 Vissza: maxIndex;  
Algoritmus vége.

### **6. feladat: void** **osszegezNaponkent**()

* + 1. Már előzőleg elvégzett felösszegzések eredményét kell csak kíirni a képernyőre, lsd.: szamolNapokat fv.

### void szamolNapokat()

Megszámoljuk a fuvarokat naponként:

Be:   
 fuvarok Fuvar tömb  
 db tömbben található elemek száma

Algoritmus:  
 Ciklus i legyen 1-től 7-ig egyesével  
 naponkentiDb[i] := 0;  
 naponkentiKm[i] := 0;  
 Ciklus vége.   
 Ciklus i := 0-tól míg kisebb mint db egyesévvel  
 naponkentiDb[fuvarok[i].nap] := naponkentiDb[fuvarok[i].nap] + 1;  
 naponkentiKm[fuvarok[i].nap] :=  
 naponkentiKm[fuvarok[i].nap] + fuvarok[i].tavolsag;  
 Ciklus vége.   
Algoritmus vége.

### int szamolOsszesenKiirReszosszeg()

Összegzés tétele alapján felösszegezzük a fuvarok távolság alapján kalkulált összegeket:

Be:   
 fuvarok Fuvar tömb  
 db tömbben található elemek száma

Algoritmus:  
 fiz := 0;  
 Ciklus i := 0-tól míg kisebb mint db egyesével  
 fiz := fiz + szamolTavAlapjan(fuvarok[i].tavolsag);  
 Ciklus vége.   
 Vissza: fiz;  
Algoritmus vége.

### int szamolTavAlapjan(int tav)

Meghatározza a járandóságot a távolság alapján:

Be:   
 tav

Algoritmus:  
 fiz := 0;  
 Ha tav kisebb mint 0 akkor fiz := 0;  
 Egyébként ha tav kisebb mint 4 akkor fiz := 500;  
 Egyébként ha tav kisebb mint 6 akkor fiz := 700;  
 Egyébként ha tav kisebb mint 11 akkor fiz := 900;  
 Egyébként ha tav kisebb mint 21 akkor fiz := 1400;  
 Egyébként ha tav kisebb mint 31 akkor fiz := 2000;  
 Vissza: fiz;  
Algoritmus vége.

# Felhasználói dokumentáció

### Hardver és szoftver követelmények:

Minimális hardverigény:

* 1 GHz-es vagy gyorsabb processzor
* 512 MB RAM 32 bites vagy 1GB RAM 64 bites rendszer
* 1 MB szabad hely a program és az adatfájlok tárolására
* 800\*600 pixel felbontású kijelző

Ajánlott hardverigény:

* 1 GHz vagy gyorsabb legalább 2 magos 64 bites processzor
* 4 GB RAM
* 1 MB szabad hely a program és az adatfájlok tárolására
* nagy felbontású (720p) kijelző

Szoftverigény

* Operációs rendszer: Microsoft Windows 10 / 11 vagy Linux operációs rendszer változatai
* Program futtatási környezet: C++ fordító, vagy IDE, mint pl. Visual Studio
* Fájlkezelés: a program a futtatható könyvtárban elhelyezett utak.txt fájlt használja bemenetként.
* Kimeneti fájl: A program egy befizetes.txt nevű fájlt hoz létre a kimenetek tárolására.

### Program kezelése

**Program telepítése**

A program nem igényel telepítést, amennyiben a futtatási környezet rendelkezésre áll.

**Program indítása Linux-on**

1. Biztosítsa, hogy az utak.txt fájl a programmal azonos könyvtárban található.
2. Futtassa a programot parancssorból az alábbi parancs kiadásával:   
   ./futarszolgalat

**Program indítása Windows-on**

1. Biztosítsa, hogy az utak.txt fájl a programmal azonos könyvtárban található.
2. Futtassa a programot parancssorból az alábbi parancs kiadásával: futarszolgalat.exe

**Működése és eredmények:**

1. Hét második útjának távolsága: a program kiírja a hét második útjának távolságát kilométerben.
2. Hét utolsó útjának távolsága: megjeleníti a hét utolsó fuvarjának távolságát.
3. Szabadnapok megjelenítése: a program felsorolja azokat a napokat, amelyeken a futár nem dolgozott.
4. Legtöbb fuvarral rendelkező nap: megmutatja azt a napot, amelyen a legtöbb fuvar történt.
5. Naponkénti kilométerek összesítése: napokra bontva kiírja az összes megtett távolságot.
6. Egyedi távolság díjazása: a felhasználó által megadott távolság alapján meghatározza a fizetést.
7. Heti összesítés: az összesített heti díjazást számolja ki és jeleníti meg.
8. Kimeneti fájl létrehozása: a befizetes.txt fájlban napokra bontva, csökkenő sorrendben jeleníti meg az utak díjazását.

**Felhasználói bemenet:**

1. A program az indítás után beolvassa az utak.txt fálj tartalmát.
2. A távolság alapú fizetés kiszámításához a program a felhasználótól kér egy távolság adatot km-ben.  
   (pl.: „ Adjon meg egy tavolsagot: „)

**Kimenet:**

1. A megadott távolságért járó fizetést adja meg.  
   (pl.: „Jarandosag: „ vagy „ Heti jarandosag: „)
2. A program kimenete a befizetes.txt fájlban napokra bontva található.

**Program leállítása**

Egy billentyűzet lenyomására vagy a futtató ablak bezárásával a program leáll.

### Tesztelés

Tesztadatok: Az utak.txt fájl a következő minta adatokat tartalmazza:

1 1 5

1 2 9

3 2 12

1 4 3

3 1 7

Várható eredmények:

1. Hét második útja: 9 km
2. Hét utolsó útja: 7 km
3. Szabadnapok: 2. nap, 4. nap, 5. nap, 6. nap
4. Legtöbb fuvar: 1. nap
5. Naponkénti távolságok:
6. 1. nap: 17 km
7. 2. nap: 0 km
8. 3. nap: 19 km
9. Felhasználói távolságra díjazás:
   * Bevitt érték: 7 km → Eredmény: 900 Ft
10. Heti összes fizetés: 5 500 Ft
11. Kimeneti fájl (befizetes.txt) tartalma:
12. 1. nap 4. út: 700 Ft
13. 1. nap 2. út: 900 Ft
14. 1. nap 1. út: 500 Ft
15. 3. nap 2. út: 1 400 Ft
16. 3. nap 1. út: 900 Ft

**Eredmények és hibák:**

A tesztek minden esetben az elvárt eredményt adták.

Nem várt hiba nem jelentkezett a program futtatása során

**Hibalehetőségek**

Az utak.txt fájl nem található a program indítási könyvtárában, vagy más néven szerepel.

Az utak.txt fájl nem a megadott formában tartalmazza az adatokat.

### Továbbfejlesztési lehetőségek

* **Adatbevitel fejlesztése,** a txt, fájlalapú működés helyett SQL adatbázis bevezetése a gyorsabb/megbízhatóbb adatkezelés érdekében.
* **http hozzáférés, a** program működésének integrálása egy webes alkalmazásba.
* **Részletesebb felbontás,** kimutatások készítése napokra és fuvarokra lebontva.

# Irodalomjegyzék

[1] Programfejlesztés, Bevezetés az objektum-orientált programozásba GAMF, Kecskemét

[2] A C++ programozási nyelv alapjai – Oktatási segédanyag – Dr. Pásztor Attila

[3] A C++ programozási nyelv középiskolásoknak – Pánczél István

# Forráskód

#include <iostream>

#include <fstream>

using namespace std;

struct Fuvar {

int nap;

int sorsz;

int tavolsag;

};

class Feldolgoz

{

private :

// Fuvarok tömbje

Fuvar fuvarok[100];

// hány elem van a tömben

int db = 0;

string napok [8];

int naponkentiDb[8];

int naponkentiKm[8];

void szamolNapokat()

{

// ki 0-azuk a napokat

for (int i = 1; i<=7; i++){

naponkentiDb[i] = 0;

naponkentiKm[i] = 0;

}

for (int i = 0; i<db; i++){

naponkentiDb[fuvarok[i].nap] ++;

naponkentiKm[fuvarok[i].nap] += fuvarok[i].tavolsag;

}

}

static int calcErtek(Fuvar fuvar)

{

return fuvar.nap \* 100 + fuvar.sorsz;

}

void rendez()

{

for (int i = 0; i < db; i++)

{

for (int j = i+1; j < db; j++)

{

if (calcErtek(fuvarok[j]) < calcErtek(fuvarok[i])) {

Fuvar seged = fuvarok[j];

fuvarok[j] = fuvarok[i];

fuvarok[i] = seged;

}

}

}

}

public:

Feldolgoz();

// 1. feladat

void beolvas() {

ifstream be("utak.txt");

if (be.fail()) { cout << "Fajl nem nyithato meg"; exit(1); }

int i = 0;

while(!be.eof()&&(be >> fuvarok[i].nap)) {

be >> fuvarok[i].sorsz;

be >> fuvarok[i].tavolsag;

i++;

}

db = i;

}

// 2. feladat

int hetMasodikFuvarKm() {

// kiinduló értéket felvesszük olyan nagyra, melyet nem érhet el

int ertek1 = 10000;

int ertek2 = 10000;

int i1;

int i2;

for (int i = 0; i<db; i++){

int ertek = calcErtek(fuvarok[i]);

if (ertek < ertek1) {

i1 = i;

ertek1 = ertek;

}

}

for (int i = 0; i<db; i++){

if (i1 != i ) {

int ertek = calcErtek(fuvarok[i]);

if (ertek < ertek2) {

i2 = i;

ertek2 = ertek;

}

}

}

return fuvarok[i2].tavolsag;

}

// 3. feladat

int hetUtolsoFuvarKm() {

int maxNap = 0;

int maxSorszamIndex = 0;

// megkeressük az utolsó napot

for (int i = 0; i<db; i++){

if (fuvarok[i].nap > maxNap) {

maxNap = fuvarok[i].nap;

}

}

for (int i = 0; i<db; i++){

if (fuvarok[i].nap == maxNap && fuvarok[i].sorsz > fuvarok[maxSorszamIndex].sorsz) {

maxSorszamIndex = i;

}

}

return fuvarok[maxSorszamIndex].tavolsag;

}

// 4. feladat

void hetSzabadNapjai() {

// végig megyünk a napokon

for (int n = 1; n<=7; n++){

// Eldöntés tételével vizsgáljuk, hogy szabad nap volt a bizonyos nap

int i=0;

while(i<db && fuvarok[i].nap != n) i++;

// itt ez esetben pont azt kell kírni, ha nincs elem a tömbben

if (i==db) {

cout << n <<". - "<< napok[n]<< " ";

}

}

}

// 5. feladat

int legtobbFuvarNap() {

szamolNapokat();

// minimumnak vesszük az első napot

int maxIndex = 1;

for (int i = 2; i<=7; i++){

if (naponkentiDb[i]>naponkentiDb[maxIndex]) maxIndex = i;

}

return maxIndex;

}

// 6. feladat

void osszegezNaponkent()

{

// végig megyünk a napokon

for (int n = 1; n <= 7; n++)

{

cout <<" "<< n << ". nap: " << naponkentiKm[n] << endl;

}

}

string getNapStr(int i) {

return napok[i];

}

// 8-9. feladat

int szamolOsszesenKiirReszosszeg()

{

rendez();

int fiz = 0;

ofstream ki("befizetes.txt");

if (ki.fail())

{

cout << "Fajl nem nyithato meg";

exit(1);

}

for (int i = 0; i < db; i++)

{

int reszFiz = szamolTavAlapjan(fuvarok[i].tavolsag);

ki << " " << fuvarok[i].nap << ".nap " << fuvarok[i].sorsz << ".út: " << reszFiz << " Ft" << endl;

fiz += reszFiz;

}

cout << " Sikeresen elkeszult fajl: befizetes.txt" << endl;

return fiz;

}

int szamolTavAlapjan(int tav) {

int fiz = 0;

if (tav < 1) fiz = 0;

else if (tav < 4) fiz = 500;

else if (tav < 6) fiz = 700;

else if (tav < 11) fiz = 900;

else if (tav < 21) fiz = 1400;

else if (tav < 31) fiz = 2000;

return fiz;

}

static void print( Fuvar f)

{

cout << "Nap: "<< f.nap << " ssz.: " << f.sorsz << " tav: " << f.tavolsag << endl;

}

};

Feldolgoz::Feldolgoz() {

napok[1] = "Hetfo";

napok[2] = "Kedd";

napok[3] = "Szerda";

napok[4] = "Csutortok";

napok[5] = "Pentek";

napok[6] = "Szombat";

napok[7] = "Vasarnap";

}

int main()

{

Feldolgoz fd;

cout << "1. feladat: " << endl;

cout << " Beolvasas: " << endl;

// 1. feladat

fd.beolvas();

// 2. feladat

cout << "2. feladat: " << endl;

cout << " Het masodik utja kilometerben: " << fd.hetMasodikFuvarKm() << endl;

// 3. feladat

cout << "3. feladat: " << endl;

cout << " Het utolso fuvar kilometerben: " << fd.hetUtolsoFuvarKm() << endl;

// 4. feladat

cout << "4. feladat: " << endl;

cout << " Het szabadnapjai: ";

fd.hetSzabadNapjai();

cout << endl;

// 5. feladat

cout << "5. feladat: " << endl;

int max = fd.legtobbFuvarNap();

cout << " Het melyik napjan volt a legtobb fuvar: " << max << ". nap, " << fd.getNapStr(max) << endl;

// 6. feladat

cout << "6. feladat: " << endl;

cout << " Egyes napkokon megtett kilometer: "<<endl;

fd.osszegezNaponkent();

cout << endl;

// 7. feladat

cout << "7. feladat: " << endl;

cout << " Adjon meg egy tavolsagot: ";

int tav;

cin >> tav;

cout << " Jarandosag: " << fd.szamolTavAlapjan(tav) << endl;

// 8. feladat

cout << "8. feladat: " << endl;

int osszesen = fd.szamolOsszesenKiirReszosszeg();

// 9. feladat

cout << "9. feladat: " << endl;

cout << " Heti jarandosag: " << osszesen << endl;

}