Fehér Balázs Csaba, DWLQB4
Programozás házi feladat dokumentáció, **59**-es feladat

Mellékelt fájlok:

- main.cpp
- Vector.h
- Vector_test.cpp
- Queue.h
- Queue_test.cpp
- City.h
- City.cpp
- City_test.cpp
- Map.h
- Map.cpp
- Hungary.txt
- USA_roadtrip.txt
- West_coast.txt
- Empty.txt
- test_cases.jpg
- Hungary_Eger.png
- Hungary_Banreve.png
- Hungary_Tihany.png
- USA_roadtrip_Chicago.png
- West_coast_Los_Angeles.png
- Empty.png
- Cannot_open.png

Az elkészítendő programnak egy menetrend alapján kell tájékoztatást adnia a szükséges átszállásokról.

A megoldáshoz kezdésnek megvalósítottam két tárolót osztálysablonként: egy vektort és egy FIFO-t. Előbbi egy egyszerű tömb, majd elláttam a megfelelő tagfüggvényekkel és operátorokkal. A FIFO tárolót láncolt listaként írtam meg, a szükséges tagfüggvényekkel és operátorokkal.

A városok tárolására egy City nevű osztályt készítettem, mely egy városnak a nevét és a vele szomszédos városok indexét tárolja. (ezen index értelmezését lásd a köv. bekezdésben.)

A Map osztály több célt is szolgál: elvégzi a beolvasást, melynek során egy City típusú vektorban (cities) eltárolja a térkép elemeit. A beolvasás során a szomszédos (a tesztfájlban azonos sorban lévő) városok cities-beli indexeit eltárolja az egyes városok neighbors vektorában. Ezzel felépült a térképnek egy szomszédsági listás gráf reprezentációja, amelyen a BFS algoritmus futhat. A beolvasott térkép városait printMap() tagfüggvénye kijelzi a felhasználónak. A Map osztály továbbá egy, a felhasználó által megadott városból indulva szélességi kereséssel rögzíti a városok távolságát a gyökértől, ezzel adva meg a szükséges átszállások számát. A kapott eredmény kijelzésére egy printTransfers(..) nevű függvényt írtam.

A program működése röviden: létrehozok a main.cpp-ben egy Map objektumot, melynek konstruktorában megadható a tesztfájl neve. A konstruktor meghívja a readFile(..) függvényt, amely beolvassa, és eltárolja a térképet. Ezután a térképet a Map osztály printMap() tagfüggvénye megjeleníti. A felhasználótól bekérek egy indexet. Ebből az indexű városból induló járatok átszállási számát adja meg a program. A megadott gyökérrel lefut a Map BFS(..) függvénye, majd a printTransfers(..) függvény kijelzi az eredményt.

Hiba- és kivételkezelés: a program csak érvényes indexet fogad el a felhasználótól, melyet egy egyszerű while ciklussal oldottam meg. A fájl megnyitásának sikertelensége esetén a readFile(..) kivételt dob, mellyel tájékoztatja a felhasználót a problémáról. Ha fájl megnyitása és beolvasása után az adatszerkezet üres, vagyis egy üres fájlt kap a program, akkor ugyanezen függvény bezárja a fájlt, majd kivételt dob. Így egyik esetben sem kér új adatot a felhasználótól, sem a print függvények, sem a BFS nem fut le, a program leáll. Néhány tesztesetről ernyőképet is mellékeltem.

A program tesztelése:

Mindkét tárolóhoz készítettem tesztmodult, melyben háromféle típussal teszteltem az osztályokat. A vektor osztályt a saját készítésű City osztállyal is. A tesztek eredménye az elvárt működésnek megfelelő volt. A City osztály tesztmoduljában az osztály tagfüggvényeit és feladat megoldására való alkalmasságát teszteltem. Egy korábbi verzióban a neighbors vektor a szomszédokra mutató pointereket tartalmazott, azonban ez a megoldás hibásnak bizonyult, amikor elegendően nagy térkép esetén a cities vektor megtelt, és a vektor dinamikus nyújtózkodási miatti új memóriafoglalás miatt a pointerek érvénytelenné váltak. A szomszédos városok indexének tárolása viszont megfelelőnek tűnt. A map osztály tesztelésére nem készítettem külön modult, hiszen annak helyes működéséről a már kész program tesztelésével bizonyosodhatunk meg.

A programot háromféle bemeneten, különböző kiindulási városokkal futtattam. (Hungary.txt, USA_roadtrip.txt, West_coast.txt). A kapott eredményeket a mellékelt térképpel összevetettem, valamennyi eredmény a vártnak megfelelt. Az egyes tesztesetekről ernyőképeket is mellékeltem.