

Bieszczadzki Tour

Specyfikacja funkcjonalna

Maciej Czarkowski, 07.11.2019

Spis treści

1	Wstęp	2
1.1	Cel dokumentu	2
1.2	Cel projektu	2
1.3	Użytkownik docelowy	2
2	Uruchomienie programu	3
3	Dane wejściowe	3
3.1	Plik konfiguracyjny	4
3.2	ID_miejsca	4
3.3	Lista życzeń	5
4	Dane wyjściowe	6
5	Scenariusz uruchomienia	6
6	Opis sytuacji wyjątkowych	7
7	Testowanie	7

1 Wstęp

1.1 Cel dokumentu

Celem niniejszego dokumentu, będącego specyfikacją funkcjonalną projektu „*Bieszczadzki Tour*”, realizowanego na potrzeby zadania projektowego z przedmiotu Algorytmy i Struktury Danych, jest wprowadzenie użytkownika końcowego w jego problematykę i możliwości. Ma on za zadanie również pełne poinstruowanie użytkownika, w jaki sposób odpowiednio i bezproblemowo korzystać z omawianego programu.

1.2 Cel projektu

Celem projektu jest pomoc zakłopotanym studentom w odnalezieniu optymalnej trasy pomiędzy wybranymi przez nich punktami turystycznymi na mapie Bieszczad. Stworzony program ma umożliwiać wygenerowanie trasy przez wszystkie wybrane przez studentów miejsca w możliwie najkrótszym czasie, z uwzględnieniem kosztów podróży.

1.3 Użytkownik docelowy

Użytkownikiem końcowym programu „*Bieszczadzki Tour*” jest grupa studentów wybierających się na wyprawę po Bieszczadach, którzy z wykorzystaniem programu mają możliwość zoptymalizowania swojej wędrowki. Użytkownikiem końcowym jest również mgr. Inż. Paweł Zawadzki, prowadzący zajęcia z Algorytmów i Struktur Danych w semestrze 2019Z, który to na podstawie działania programu dokona jego oceny.

2 Uruchomienie programu

W celu uruchomienia programu, na lokalnym komputerze należy otworzyć wiersz poleceń, a następnie przejść do folderu z projektem, podając ścieżkę do niego w pokazany poniżej sposób, z wykorzystaniem komendy `cd`.

```
cd C:\path\to\project\directory
```

Po przejściu do powyższego folderu należy uruchomić program, podając jednocześnie argumenty do niego, zgodnie z poniższym schematem.

```
java -jar dist\bieszczadzkitour.jar path\to\data.txt ID_miejsca path\to\wishlist.txt
```

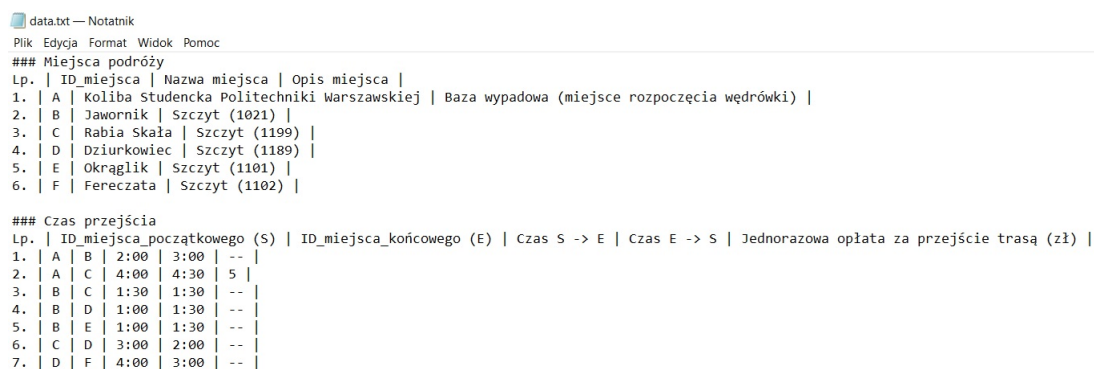
W powyższej komendzie `path\to\data.txt` to ścieżka do pliku, w którym użytkownik powinien umieścić dane wejściowe dla programu (nazywanego dalej plikiem konfiguracyjnym), `ID_miejsca` to identyfikator miejsca rozpoczęcia podróży, zgodny z informacjami zawartymi w pliku konfiguracyjnym, a opcjonalnie dodajemy `path\to\wishlist.txt`, czyli ścieżkę do pliku, w którym użytkownik określa miejsca, które chce odwiedzić (nazywanego dalej listą życzeń). Formatowanie danych wejściowych dla programu jest omówione szczegółowo w punkcie trzecim niniejszego dokumentu.

3 Dane wejściowe

Danymi wejściowymi dla programu są wspomniane w poprzednim punkcie argumenty, które podajemy przy uruchamianiu. Pliki, będące argumentami do programu, powinny być plikami tekstowymi, które muszą posiadać określone formatowanie zawartości, aby program mógł działać w pełni poprawnie. W poniższych podsekcjach bardziej szczegółowo omówiono te elementy.

3.1 Plik konfiguracyjny

W tym pliku tekstowym (należy wykorzystać rozszerzenie `.txt`) użytkownik powinien umieścić wszelkie informacje (unikatowy numer ID, nazwę oraz opis miejsca) na temat punktów, które rozpatrujemy podczas działania programu oraz informacje na temat czasów (w obu kierunkach) i cen przejść pomiędzy poszczególnymi punktami. Plik powinien być zgodny z poniższym schematem.



```
data.txt — Notatnik
Plik  Edycja  Format  Widok  Pomoc

### Miejsca podróży
Lp. | ID_miejsca | Nazwa miejsca | Opis miejsca |
1. | A | Koliba Studencka Politechniki Warszawskiej | Baza wypadowa (miejsce rozpoczęcia wędrowki) |
2. | B | Jawornik | Szczyt (1021) |
3. | C | Rabia Skała | Szczyt (1199) |
4. | D | Dziurkowiec | Szczyt (1189) |
5. | E | Okraglik | Szczyt (1101) |
6. | F | Fereczata | Szczyt (1102) |

### Czas przejścia
Lp. | ID_miejsca_początkowego (S) | ID_miejsca_końcowego (E) | Czas S -> E | Czas E -> S | Jednorazowa opłata za przejście trasą (zł) |
1. | A | B | 2:00 | 3:00 | -- |
2. | A | C | 4:00 | 4:30 | 5 |
3. | B | C | 1:30 | 1:30 | -- |
4. | B | D | 1:00 | 1:30 | -- |
5. | B | E | 1:00 | 1:30 | -- |
6. | C | D | 3:00 | 2:00 | -- |
7. | D | F | 4:00 | 3:00 | -- |
```

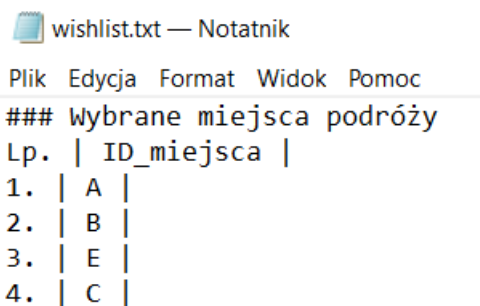
Rysunek 1: Poprawne formatowanie pliku z danymi wejściowymi

3.2 ID_miejsca

Jest to ciąg znaków, zgodny z analogicznym polem w pliku konfiguracyjnym, który powinien jednoznacznie identyfikować i określać miejsce rozpoczęcia i jednocześnie zakończenia podróży.

3.3 Lista życzeń

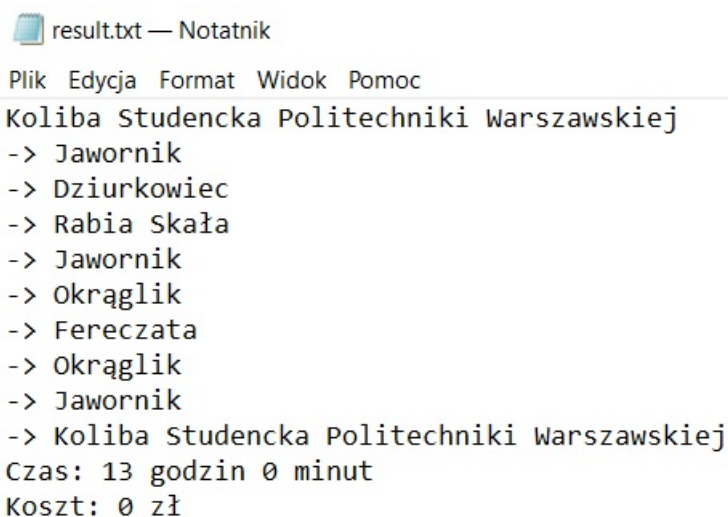
W tym pliku tekstowym (ponownie należy wykorzystać rozszerzenie `.txt`) użytkownik określa jednoznacznie, za pomocą pola `ID_miejsca`, miejsca, które chce odwiedzić. Formatowanie pliku powinno być zgodne z poniższym schematem.



Rysunek 2: Poprawne formatowanie pliku z listą życzeń

4 Dane wyjściowe

W wyniku działania programu tworzony jest tekstowy plik wynikowy o nazwie `result.txt`, w którym przedstawiona jest pełna, najkrótsza i możliwie najtańsza trasa z wyliczonym czasem oraz kosztem wędrowki. Plik tworzony będzie w folderze projektu, w podfolderze `output`, a jego formatowanie będzie zgodne z poniższym schematem.



```
result.txt — Notatnik
Plik Edycja Format Widok Pomoc
Koliba Studencka Politechniki Warszawskiej
-> Jawornik
-> Dziurkowiec
-> Rabia Skała
-> Jawornik
-> Okrąglik
-> Fereczata
-> Okrąglik
-> Jawornik
-> Koliba Studencka Politechniki Warszawskiej
Czas: 13 godzin 0 minut
Koszt: 0 zł
```

Rysunek 3: Formatowanie pliku wynikowego

5 Scenariusz uruchomienia

Po uruchomieniu programu zgodnie z punktem 2 niniejszej specyfikacji, w przypadku poprawnego formatowania plików, działanie programu zakończy się, a w podfolderze `.\output` w folderze projektowym ukaże się nowy plik wynikowy `result.txt`.

6 Opis sytuacji wyjątkowych

W przypadku podania nieprawidłowych argumentów przy uruchamianiu programu, w powłoce tekstowej, w której uruchamiany jest program, wyświetlony zostanie stosowny komunikat o niepoprawnym sformatowaniu argumentów z informacją, w którym argumencie wystąpił błąd. Dzięki tej informacji, użytkownik może poprawić wadliwy argument i ponownie przystąpić do uruchomienia programu.

7 Testowanie

Program będzie testowany poprzez podawanie jako argumenty różnych zestawów danych – zarówno poprawnych, jak i niepoprawnych. W przypadku podania danych o niepoprawnym formatowaniu sprawdzana będzie funkcjonalność w jaki sposób program informuje nas o błędach w argumentach – czy komunikat o błędzie jest wyświetlony prawidłowo oraz czy jednoznacznie informuje nas jaki błąd wystąpił.

W przypadku testowania poprawnego działania programu, przygotowane zostaną zestawy danych wejściowych o poprawnym formatowaniu oraz o wiadomym poprawnym wyniku działania programu. Po wykonaniu programu, dane z pliku wynikowego zostaną porównane z wynikiem oczekiwanym, co pozwoli nam stwierdzić czy program zadziałał prawidłowo.