VaccDistributor

Kamil Sztandur

07/11/2020

Rozdział 1: OPIS OGÓLNY

Podrozdział 1.1: Nazwa programu.

Program nazywa się **VaccDistributor**.

Podrozdział 1.2: Poruszany problem.

Problemem rozwiązywanym przez ten program jest zagadnienie jak najbardziej optymalnego pod kątem ekonomicznym rozdysponowania szczepionek pomiędzy poszczególnymi dostawcami, a aptekami, na podstawie podanych przez użytkownika połączeń. Program skupia się na tym, aby koszt dystrybucji szczepionek był jak najmniejszy. Nie gwarantuje, że zapotrzebowanie każdej apteki zostanie zaspokojone. Także jest to wariant egoistyczny.

Podrozdział 1.3: Użytkownik docelowy.

Użytkownikiem naszego programu powininna być grupa menadżerska odpowiedzialna za dystrybucję szczepionek. Program stanowi proste rozwiązanie jak najbardziej optymalnego ekonomicznie rozwiązania problemu.

Rozdział 2: OPIS FUNKCJONALNOŚCI

Podrozdział 2.1: Jak korzystać z programu?

Uruchomienie programu ze względu na brak oprawy graficznej może sprawiać trudności mniej wtajemniczonym w komputery użytkownikom. Program uruchamia się z poziomu terminalu systemu operacyjnego, na którym jest zainstalowana najnowsza wersja oprogramowania Java.

Podrozdział 2.2: Uruchomienie programu.

Aby uruchomić program wystarczy wpisać komendę w terminalu systemu operacyjnego spełniającego opisane wymagania:

```
java -jar vaccDistributor.jar input.txt
```

znajdując się w katalogu, w którym umieszczone zostały równocześnie plik programu **vaccDistributor.jar** oraz tekstowy plik wejściowy z danymi (tutaj **input.txt**). Należy pamiętać o dopisaniu rozszerzenia pliku. Sama nazwa (tutaj **input**) nie pozwoliłaby na odnalezienie wskazanego pliku.

Należy dokładnie przestudiować strukturę pliku wejściowego na podstawie poniższego przykładu, ponieważ wszelkie odstępstwa od ogólnoprzyjętego szablonu mogą spowodować nieprawidłową pracę, a nawet zatrzymanie pracy programu. Tekstowy plik wejściowy powinien zawierać trzy nagłówki, oznaczające kolejno listy producentów, aptek i dostępnych połączeń. Kolejne parametry w linijce powinny być oddzielna ciągiem znaków " | ". Nie należy powielać nagłówków ani umieszczać ich w innej kolejności.

Podrozdział 2.3: Przykładowy plik wejściowy.

```
# Producenci szczepionek (id — nazwa — dzienna produkcja)
0 | BioTech 2.0 | 900
1 | EkoPolska 2020 | 1300
2 | Post-Covid Sp. z o.o. | 1100
# Apteki (id | nazwa | dzienne zapotrzebowanie)
0 | CentMedEko Centrala | 450
1 | CentMedEko 24h | 690
2 | CentMedEko Nowogrodzka | 1200
# Połączenia producentów i aptek (id producenta | id apteki | dzienna maksymalna liczba dostarczanych szczepionek | koszt szczepionki [zł] )
0 | 0 | 800 | 70.5
0 | 1 | 600 | 70
0 | 2 | 750 | 90.99
1 | 0 | 900 | 100
1 | 1 | 600 | 80
1 | 2 | 450 | 70
2 | 0 | 900 | 80
2 | 1 | 900 | 90
2 | 2 | 300 | 100
```

Rozdział 3: FORMATY DANYCH I STRUKTURA PLIKÓW.

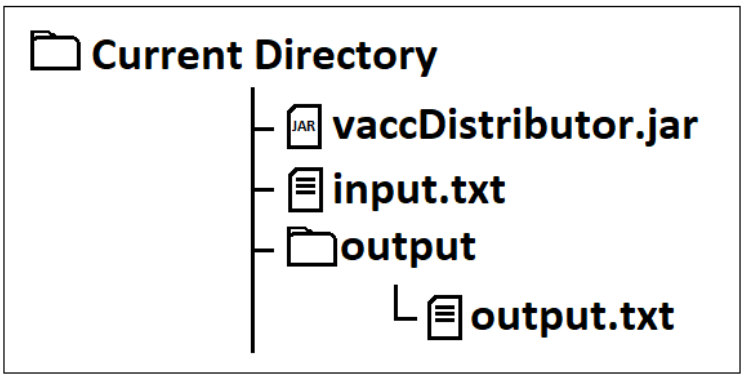
Podrozdział 3.1: Pojęcia i obiekty dziedziny aplikacji (słownik dziedziny).

1. **Producent (in. dostawca, "supplier"):**
   - **ID** - numer identyfikacyjny producenta na liście. Musi być unikatowy.
   - **Nazwa ("name")**- nazwa producenta. Musi być unikatowa.
   - **Dzienna produkcja ("daily production")** - Ilość dostępnych codziennie szczepionek gotowych do dystrybucji do aptek.
   - **Dostępna dzienna produkcja ("available daily production")** - aktualna ilość dostępnych szczepionek gotowych do dystrybucji do aptek. Od poprzedniego parametru różni się tym, że uwzględnia już podpisane umowy na dowóz określonej ilości do niektórych placówek.
2. **Apteka (in. placówka, "pharmacy"):**
   - **ID** - numer identyfikacyjny apteki na liście. Musi być unikatowy.
   - **Nazwa ("name")**- nazwa placówki apteki. Musi być unikatowa.
   - **Dzienne zapotrzebowanie ("daily production")** - ilość szczepionek, która musi być dostarczana dziennie, aby zaspokoić potrzeby placówki.
   - **Obecne dziennie zapotrzebowanie ("current daily need")** - aktualna ilość pozostałych szczepionek, która musi być dostarczana dziennie, aby zaspokoić potrzeby placówki. Uwzględnia umowy już podpisane na dostarczenie pewnej ilości i odejmuje je od domyślnego dziennego zapotrzebowania.
3. **Połączenie ("connection"):**
   - **ID dostawcy ("supplier ID")** - numer identyfikacyjny dostawcy, od którego wychodzi dane połączenie.
   - **ID apteki ("pharmacy ID")** - numer identyfikacyjny apteki, do której dochodzi dane połączenie.
   - **Maksymalny transfer ("max transfer")** - maksymalna liczba szczepionek, która może zostać przewieziona jednego dnia tym połączeniem.
   - **Pozostały transfer ("available transfer")** - pozostała liczba szczepionek, która może zostać przewieziona jednego dnia tym połączeniem. Uwzględnia już podpisane umowy na dowożenie tym połączeniem pewnej ilości szczpeionek.
   - **Koszt pojedynczej szczepionki ("cost per single vaccine")** - koszt dostarczenia pojedynczej szczepionki tym połączeniem.
4. **Transakcja ("transaction").** Transakcja jest połączeniem o parametrach, które program uznał za najbardziej możliwie optymalne. W przeciwieństwie do listy wszystkich połączeń, transakcja jest połączeniem, które na pewno zostanie zrealizowane i jest opracowywane w trakcie pracy programu:
   - **ID dostawcy ("supplier ID")** - numer identyfikacyjny dostawcy, od którego wychodzi dane połączenie.
   - **ID apteki ("pharmacy ID")** - numer identyfikacyjny apteki, do której dochodzi dane połączenie.

- **Transfer ("max transfer")** - ilość szczepionek, która zostanie dostarczona tym połączeniem.
- **Koszt pojedynczej szczepionki ("cost per single vaccine")** - koszt dostarczenia pojedynczej szczepionki tym połączeniem.
- **Całkowity koszt ("total cost")** - całkowity koszt wykonania tej transakcji:

$$\textbf{Całkowity koszt} = \textbf{Koszt pojedynczej szczepionki} \times \textbf{Transfer}$$

Podrozdział 3.2: Struktura katalogów.



Plik wejściowy z danymi producentów, aptek i połączeń (tutaj **input.txt** powinien znajdywać się w tym samym katalogu co plik wykonywalny programu (tutaj **vaccDistributor.jar**). Po uruchomieniu programu zgodnie z instrukcjami opisanymi w poprzednim rozdziale zostanie utworzony podkatalog o nazwie **output**, w którym znajdzie się plik z optymalnym rozwiązaniem problemu **output.txt**.

Podrozdział 3.3: Przechowywanie danych w programie.

PRZYPOMNIENIE Poprawne wywołanie programu:

```
java -jar vaccDistributor.jar input.txt
```

Zgodnie z poprawnym wywołaniem i przebiegem pracy programu powinien wyświetlić się taki komunikat. Symbolizować on będzie zakończenie pracy programu i pomyślne umieszczenie pliku z rozwiązaniem w podkatalogu output:

```
home:$ java -jar vaccDistributor.jar input.txt
Czytanie danych wejściowych... Sukces.
Znajdywanie najlepszych połączeń... Sukces.
Tworzenie katalogu z rozwiazaniem... Sukces.
Zapisywanie danych wyjściowych do pliku... Sukces.

Praca programu zakończona pomyślnie.
```

W przypadku nie podania żadnego argumentu zostanie wyświetlony komunikat informujący o prawidłowym wywołaniu programu:

```
home:$ java -jar vaccDistributor.jar
Brak podanego pliku z danymi. Poprawne wywołanie:
        java -jar vaccDistributor.jar input.txt
gdzie input.txt to nazwa pliku z danymi, który znajduje się w bieżącym katalogu.

Praca programu zakończona z błędem.
```

W przypadku kiedy niewystarczająca wydajność producentów lub połączeń z aptekami nie jest w stanie zaspokoić ich wszystkich zostanie wyświetlony stosowny komunikat.

```
home:$ java -jar vaccDistributor.jar input.txt
Czytanie danych wejściowych... Sukces.
[Ostrzeżenie] Producenci nie są dostatecznie wydajni dla tak wymagających aptek.
Znajdywanie najlepszych połączeń... Sukces.
Tworzenie katalogu z rozwiazaniem... Sukces.
Zapisywanie danych wyjściowych do pliku... Sukces.

Praca programu zakończona pomyślnie.
```

W przypadku podania nieistniejącego pliku zostanie wyświetlony stosowny komunikat:

```
home:$ java -jar vaccDistributor.jar input.txt
Czytanie danych wejściowych... Porażka. Podany plik nie istnieje.

Praca programu zakończona z błędem.
```

W przypadku podania nieprawidłowego pliku zostaną wyświetlony komunikat zawierający przybliżony opis błędu:

```
home:$ java -jar vaccDistributor.jar input.txt
Czytanie danych wejściowych... Porażka. Podany plik jest nieprawidłowy.
Opis błędu: Cost per single vaccine cannot be less than 0 at line 49.

Praca programu zakończona z błędem.
```

Podrozdział 3.4: Przykładowy plik wyjściowy.

Plik wejściowy zawiera listę zleconych transakcji w formacie:

NAZWA PRODUCENTA -> NAZWA APTEKI [Koszt = LICZBA SZCZEPIONEK * KOSZT POJEDYNCZEJ SZCZEPIONKI = CALKOWITY KOSZT]

Dodatkowo na końcu widnieje suma kosztów wszystkich transakcji poprzedzona stosowną etykietą **"Opłaty całkowite: "** po której zapisana zostaje suma kosztów wszystkich transakcji.

Przykładowy plik wyjściowy:

BioTech 2.0 -> CentMedEkoCentrala [Koszt = 300 * 70.5 = 21150 zł]
EkoPolska 2020 -> CentMedEkoCentrala [Koszt = 150 * 100 = 15000 zł]

/*...kolejne wiersze opisujące ustalone połączenia pomiędzy producentami, a aptekami...*/

Opłaty całkowite: 36150 zł

Rozdział 4: SCENARIUSZ DZIAŁANIA PROGRAMU.

Podrozdział 4.2: Scenariusz ogólny.

1. Przeczytanie wskazanego przez użytkownika pliku z danymi.
2. Obliczenie i utworzenie najbardziej optymalnych transakcji producenci - apteki.
3. Wypisywanie listy transakcji do pliku tesktowego w folderze **output**.

Podrozdział 4.2: Scenariusz szczegółowy.

1. Sprawdzenie, czy użytkownik podał jakiś plik wejściowy.
   - Jeżeli nie, poinformuj go o tym, pokaż przykład prawidłowego wywołania i zakończ działanie.
2. Sprawdzenie, czy plik podany przez użytkownika istnieje.
   - Jeżeli nie, poinformuj go o tym i zakończ działanie.
3. Czytanie podanego pliku. Zczytywanie kolejnych producentów, aptek i połączeń do wewnętrznych kontenerów. Sprawdzanie składni na bieżąco.
   - Jeżeli składnia jest nieprawidłowa powiadom użytkownika, wskaż nr. linii i zakończ pracę.
   - Jeżeli występują duplikaty, pomiń linię, wskaż jej numer i powiadom użytkownika potokiem błędu. Kontynuuj pracę programu.
4. Oblicz, czy istnieją apteki, które nie mogą zostać w żaden sposób zaspokojone.
   - Jeżeli tak, to powiadom użytkownika o tej sytuacji. Kontynuuj pracę programu i zapamiętaj ten fakt dla algorytmu.
5. Rozpocznij działanie algorytmu, tworzącego jak najtańsze połączenia dostawcy - apteki:
   - Posortuj quicksortem listę połączeń rosnąco pod względem kosztu pojedynczej szczepionki.
   - Duplikaty posortuj quicksortem malejąco pod względem rzeczywistego transferu. Za rzeczywisty transfer przyjmij najmniejszą liczbę z:
     - aktualne zapotrzebowanie apteki danego połączenia
     - aktualnie dostępna liczba szczepionek u producenta danego połączenia
     - aktualna przepustowość połączenia
   - Realizuj kolejne połączenia z początku listy w transakcje. W przypadku wyczerpania producenta, połączenia lub zaspokojenia apteki skreślaj z listy wszystkie pozycje je zawierające.
   - Zwróć listę transakcji.
6. Utwórz katalog output.
7. Utwórz plik output.txt w katalogu output i zapisz do niego zgodnie z szablonem listę utworzonych transakcji.
8. Poinformuj użytkownika o pomyślnym przebiegu programu. Zakończ działanie.

Podrozdział 4.3: Ekrany działania programu.

Program przebiega w pełni w terminalu systemu operacyjnego, a na końcu wypisuje plik tekstowy w sposób zgodny z opisanym w poprzednich rozdziałach. Przykładowy ekran pracy programu wygląda w ten sposób:

Każde bieżąco wykonywane zadanie sygnalizowane jest opisem kończącym się trzema kropkami "..." i tekst pozostanie w takim stanie na ekranie dopóki nie zostanie ono zakończone lub przerwane z powodu błędu. Wówczas zaraz po trzech kropkach zostanie wypisany stan zadania.

```
home:$ java -jar vaccDistributor.jar input.txt
Czytanie danych wejściowych...
```

```
home:$ java -jar vaccDistributor.jar input.txt
Czytanie danych wejściowych... Sukces.
Znajdywanie najlepszych połączeń... Sukces.
Tworzenie katalogu z rozwiazaniem... Sukces.
Zapisywanie danych wyjściowych do pliku... Sukces.

Praca programu zakończona pomyślnie.
```

```
home:$ java -jar vaccDistributor.jar input.txt
Czytanie danych wejściowych... Porażka. Podany plik jest nieprawidłowy.
Opis błędu: Cost per single vaccine cannot be less than 0 at line 49.

Praca programu zakończona z błędem.
```

Rozdział 5: TESTOWANIE.

Podrozdział 5.1: Ogólny przebieg testowania.

NAZWA TESTU	OPIS TESTU	SPOSÓB WERYFIKACJI
Brak pliku wejściowego	Sprawdzenie, czy program poinformuje użytkownika o poprawnym użyciu w przypadku braku podania pliku wejściowego.	Uruchomienie programu bez podawania żadnych argumentów.
Sprawdzenie poprawnej liczby argumentów w pliku z danymi.	Monitorowanie, czy czytania linia zawiera niewłaściwą liczbę argumentów.	Podanie programowi pliku, w którym któraś linia będzie zawierała zbyt dużo argumentów i pliku, w którym zbyt mało.
Sprawdzenie poprawnego oddzielania danych	Monitorowanie, czy czytania linia zawiera właściwy rozdział danych.	Podanie programowi pliku, w którym zamiast " | " dane oddzielane są jakimkolwiek innym sposobem.
Sprawdzenie poprawności danych numerycznych	Monitorowanie, czy czytania linia zawiera niewłaściwe dane numeryczne.	Podanie programowi pliku, w którym zamiast liczby np. szczepionek podamy jakieś słowo.
Wykrywanie i reagowanie na duplikaty	Zweryfikowanie, czy program pominie duplikaty producentów/aptek/połączeń w pliku.	Podanie programowi pliku, w którym występują duplikaty producentów/aptek/połączeń i sprawdzenie, czy poinformuje on w potoku błędów o duplikacie, a nastepnie na podstawie pliku wyjściowego, czy je pominie.
Sprawdzenie logicznej poprawności danych numerycznych	Monitorowanie, czy czytania linia nie zawiera ujemnych liczb tam.	Podanie programowi pliku, w którym koszt szczepionki byłby ujemny.
Niemożliwa do zaopatrzenia apteka	Sprawdzenie, czy program wykryje i zareaguje na sytuację, gdy danej apteki nie da się zaopatrzyć.	Podanie programowi takich danych, aby któraś z aptek nie była w stanie być zaopatrzona.
Wydajność algorytmu.	Sprawdzenie, czy jakość algorytmu oferowanego przez program jest dobra.	Zlecenie rozwiązania takiego samego problemu ekspertowi w dziedzinie analityki finansowej i temu programowi. Porównanie wyników.
Weryfikacja pliku wyjściowego	Sprawdzenie, czy plik wyjściowy jest zgodny z przyjętymi standardami.	Uruchomienie programu dla pseudolosowych danych i zweryfikowanie formatu pliku wyjściowego.
Weryfikacja oprawy wizualnej	Sprawdzenie, czy przeciętny użytkownik odnajdzie się w programie.	Pokazanie programu niezwiązanemu z nim użytkownikowi i sprawdzenie, czy potrafi z niego skorzystać bez większych trudności.