Laboratorium

Programowania Komputerów

# Temat: Biuro podróży

Autor: Alicja Lachman

Informatyka, semestr 3, grupa 1

Prowadząca: dr inż. Karolina Nurzyńska

1. **Temat**

Implementacja programu do obsługi biura podróży w języku C++.

1. **Analiza, projektowanie**

Biuro podróży obsługuje wycieczki objazdowe oraz wczasy. Dodatkowo istnieje możliwość łączenia wycieczki objazdowej z wczasami rozpoczynającymi się po jej zakończeniu w tym samym kraju. Oferta biura podróży generowana jest do pliku wyjściowego (jego nazwa podawana jest w linii poleceń jako argument programu) na podstawie ofert zawartych w podanym przez użytkownika folderze z ofertami. W zależności od wybranego rodzaju wycieczki, biuro podróży szuka wycieczek objazdowych, wczasów, wycieczek łączonych lub dowolnych.

Działanie programu jest następujące:

- Program sprawdza poprawność argumentów wywołania programu

- Odczytuje kolejno wszystkie pliki z ofertami znajdujące się w podanym folderze

- W zależności od tego, czy w danym pliku znajduje się wycieczka objazdowa („WO:” na początku pliku) czy wczasy („WP:” na początku pliku), wczytana wycieczka zostaje dodana odpowiednio do wektora wczasów lub wektora wycieczek objazdowych.

- Program sprawdza w argumentach programu, jakiego typu wycieczki chce szukać użytkownik. W przypadku wczasów (-r p) przeszukany zostaje wektor lista\_wczasow pod kątem zadanych kryteriów (data, czas trwania, maksymalna cena). Po znalezieniu wczasów sprawdzana jest również możliwość zwielokrotnienia danego turnusu. W przypadku wycieczek objazdowych (-r o) przeszukany zostaje wektor lista\_objazdowek. Dla wycieczek dowolnych (-r d) przeszukane zostają kolejno oba wektory. Jeśli użytkownik wybierze wycieczkę łączoną (-r l), szukanie rozpoczyna się od znalezienia wycieczki objazdowej spełniającej dane kryteria. Następnie dla każdej znalezionej wycieczki objazdowej szukane są wczasy rozpoczynające się w tym samym kraju co ostatni punkt wycieczki objazdowej, a także spełniające kryterium daty i ceny.

1. **Specyfikacja zewnętrzna**
   1. **Obsługa programu**

Program uruchamiany jest z linii poleceń z wykorzystaniem następujących przełączników:

-k katalog z plikami z ofertą (katalog/)

-o plik wyjściowy (plik.txt)

-r rodzaj wycieczki (o – objazdowa, p – wczasy, l – łączona, d – dowolna)

-t data, od której rozpoczyna się poszukiwanie (RRRRMMDD)

-d długość wycieczki

-c maksymalna cena

Kolejność przełączników jest dowolna. Po uruchomieniu programu program wyszukuje wycieczki spełniające zadane kryteria. W przypadku wystąpienia błędów, na ekranie wypisywane są odpowiednie komunikaty.

* 1. **Format danych wejściowych**

Program wczytuje z argumentów uruchomienia programu scieżkę do folderu, w którym znajdują się pliki tekstowe z ofertami. Każda oferta znajduje się w osobnym pliku. Przyjęto następujący format danych zgodny z założeniami projektu:

Wycieczka objazdowa:

WO: Nazwa Rok.Miesiąc.Dzień Rok.Miesiąc.Dzień Miasto[Państwo],(….) Cena ŚrodekTransportu

Przykładowy plik:

WO: Objazdowka 2015.06.07 2015.06.21 Warszawa[Polska], Malbork[Polska], Berlin[Niemcy] 3000 Autokar

Wczasy:

WP: Nazwa Rok.Miesiąc.Dzień DługośćTurnusu Miasto Państwo KosztWycieczki ŚrodekTransportu JegoCena (…)

Przykłądowy plik:

WP: Egzotyczne 2015.09.12 15 Havana Kuba 4500 Samolot 3000

* 1. **Komunikaty**

Program komunikuje się z użytkownikiem za pomocą komunikatów wyświetlanych w konsoli.

W pierwszej kolejności sprawdzana jest poprawność wprowadzonych argumentów wywołania programu. W przypadku nieprawidłowej ilości argumentów wyświetlany jest komunikat

Zla ilosc argumentow wywolania programu!! Nastepuje zamkniecie programu! oraz następuje zamknięcie programu.

Po poprawnym zakończeniu pracy programu na ekranie wyświetla się następujący komunikat:

Oferta biura podrozy zostala wygenerowana i znajduje sie w pliku: (nazwa pliku)

Znaleziono x wycieczek spelniajacych dane kryteria.

Podczas wczytywania wycieczek z plików program sprawdza poprawność zawartych w nich danych. W przypadku wykrycia niepoprawnych danych, które mogłyby wpłynąć na poprawność działania programu (na przykład zły format daty czy też cena wycieczki podana w postaci słownej), dana wycieczka zostaje pominięta a na ekranie zostaje wyświetlony odpowiedni komunikat, na przykład:

Wykryto plik z nieznanym typem wycieczki

wykryto blad w cenie jednych z wczasow!

Zla cena ktoregos ze srodkow transportu! Te wczasy zostana pominiete!

Zly rodzaj transportu! Te wczasy zostana pominiete!

Zly format daty lub dlugosc turnusu jednych z wczasow. Te wczasy zostana pominiete!

Zly format daty! Ta objazdowka zostaje pominieta!

zly rodzaj transportu. Ta objazdowka zostaje pominieta!

1. **Specyfikacja wewnętrzna**
   1. **Hierarchia klas**
   2. **Opis klas**

|  |  |  |
| --- | --- | --- |
| Klasa | Rodzaj | Opis |
| BiuroPodrozy | - | Klasa obsługująca pracę biura podróży. Umożliwia odczyt ofert z plików, wyszukiwanie wycieczek o zadanych kryteriach oraz generowanie pliku z ofertami. Jest klasą zaprzyjaźnioną klas Wycieczka, Wczasy i WycieczkaObjazdowa. |
| Wycieczka | Klasa bazowa | Klasa bazowa dla poszczególnych rodzajów wycieczek. Zawiera pola wspólne dla klas potomnych opisujące nazwę wycieczki, datę jej rozpoczęcia oraz jej koszt. |
| Wczasy | Klasa potomna | Klasa opisująca wczasy. |
| WycieczkaObjazdowa | Klasa potomna | Klasa opisująca wycieczki objazdowe. |

* 1. **Klasa BiuroPodrozy**

|  |  |
| --- | --- |
| **Metoda** | **Opis** |
| void czytajPliki(char \*folder); | Funkcja czytająca kolejne pliki znajdujące się w podanym folderze. |
| void interpretujDane(string zawartosc); | Funkcja sprawdzająca, czy wczytany plik zawiera opis wczasów czy wycieczki objazdowej. |
| void tworzWczasy(vector <char\*> opis); | Funkcja tworząca nowe wczasy i dodająca je do wektora lista\_wczasow. Na podstawie przekazanego z pliku opisu wypełniane są kolejne pola wczasów. |
| void tworzObjazdowke(vector <char\*> opis); | Funkcja tworząca nową objazdówkę i dodająca ją do wektora lista\_objazdowek. Na podstawie przekazanego z pliku opisu wypełniane są kolejne pola objazdówki. |
| void szukajObjazdowek(struct tm data, int dlugosc, float cena, char przelacznik); | Funkcja przeszukująca wektor objazdówek w poszukiwaniu wycieczek spełniających podane kryteria. Przełącznik decyduje, czy szukane są tylko wycieczki objazdowe, czy również łączone. |
| int podajIlosc(); | Funkcja zwracająca ilość znalezionych wycieczek spełniających zadane kryteria. |
| void rodzajTransportu(vector<char\*>opis,int i, Wczasy &wczasy); | Funkcja sprawdzająca jakie środki transportu są do wyboru dla danych wczasów oraz wprowadzająca ich ceny do danych wczasów. |
| void rodzajTransportu2(vector<char\*>opis,int i, WycieczkaObjazdowa &objazdowka); | Funkcja sprawdzająca jaki środek transportu posiadana dana objazdówka i zapisująca tę informację do danej objazdówki. |
| void drukujWczasy(char \*sciezka, Wczasy &wczasy,float cena, int mnoznik=1); | Funkcja drukująca do pliku wyjściowego wczasy spełniające zadane kryteria. |
| void drukujObjazdowke(char \*sciezka, WycieczkaObjazdowa &objazd); | Funkcja drukująca do pliku wyjściowego wycieczki objazdowe spełniające zadane kryteria. |
| void sprawdzLaczona(WycieczkaObjazdowa &objazd, Wczasy &wczasy, int dlugosc, float cena); | Funkcja sprawdzająca czy znalezione wczasy zaczynające się w ostatnim państwie wycieczki objazdowej spełniają również kryterium daty, długości wycieczki oraz ceny. |
| void drukujLaczona(char \*sciezka, WycieczkaObjazdowa &objazd, Wczasy &wczasy); | Funkcja drukująca do pliku wyjściowego wycieczki łączone spełniające zadane kryteria. |
| bool porownajDate(struct tm data1, struct tm data2); | Funkcja sprawdzająca, czy data1 jest większa lub równa data2. |
| int obliczIloscDni(struct tm data1, struct tm data2); | Funkcja obliczająca czas w dniach pomiędzy dwiema datami. |

|  |  |
| --- | --- |
| **Zmienna** | **Opis** |
| vector <Wczasy> lista\_wczasow; | Wektor obiektów typu Wczasy, zawiera wszystkie wczasy wczytane z plików wejściowych |
| vector <WycieczkaObjazdowa> lista\_objazdowek; | Wektor obiektów typu WycieczkaObjazdowa, zawiera wszystkie objazdówki wczytane z plików wejściowych |
| Int liczbaWycieczek | Zawiera liczbę wszystkich wycieczek spełniających zadane kryteria |

* 1. **Klasa Wycieczka**

|  |  |
| --- | --- |
| **Zmienna** | **Opis** |
| char nazwa[30]; | Zawiera nazwę wycieczki |
| struct tm data\_rozpoczecia; | Zawiera datę rozpoczęcia wycieczki |
| float kosztWycieczki; | Zawiera koszt wycieczki |

* 1. **Klasa Wczasy**

|  |  |
| --- | --- |
| **Zmienna** | **Opis** |
| float koszt\_autokar; | Zawiera koszt dojazdu autokarem |
| float koszt\_samolot; | Zawiera koszt dojazdu samolotem |
| float koszt\_wlasny; | Zawiera koszt dojazdu własnego |
| int dlugosc\_turnusu; | Informuje o długości turnusu |
| char destynacja\_kraj[30]; | Przechowuje informację o kraju wczasów |
| char destynacja\_miasto[30]; | Przechowuje informację o mieście wczasów |

* 1. **Klasa WycieczkaObjazdowa**

|  |  |
| --- | --- |
| **Zmienna** | **Opis** |
| char dojazd[30]; | Zawiera informację o rodzaju transportu |
| struct tm data\_zakonczenia; | Zawiera datę zakończenia wycieczki |
| vector <string> lista\_miast; | Wektor zawierający nazwy miast odwiedzonych podczas wycieczki |
| set <string> lista\_krajow; | Zbiór (niepowtarzających się) krajów odwiedzonych podczas wycieczki |
| string ostatni\_kraj; | Nazwa ostatniego odwiedzonego kraju, służy do porównywania z krajem wczasów podczas poszukiwania wycieczek łączonych |

1. **Testowanie i uruchamianie**

Program został przetestowany dla dużej ilości plików z ofertami. Sprawdzano również działanie programu w przypadku błędnego formatu danych zawartych w niektórych plikach. Dzięki zastosowaniu odpowiednich zabezpieczeń program właściwie reaguje na błędne dane i nie zawiesza się, a jedynie komunikuje wystąpienie błędu i w zależności od jego wagi kontynuuje działanie lub informuje użytkownika o zakończeniu działania.

//Program został również sprawdzony pod kątem wycieków pamięci przy użyciu narzędzia Valgrind w środowisku Linux. Nie wykryto żadnych wycieków pamięci.