Politechnika Śląska w Gliwicach
Wydział Automatyki, Elektroniki i Informatyki

Podstawy Programowania Komputerów

Projekt - loty

autor	Marcin Strządała
prowadzący	dr inż. Wojciech Sułek
rok akademicki	2018/2019
kierunek	teleinformatyka
rodzaj studiów	SSI
semestr	1
termin laboratorium	środa, 10:15 – 11:45
grupa	1
sekcja	2
termin oddania sprawozdania	2019-01-16
data oddania sprawozdania	2019-01-22

1 Treść zadania

Napisać program, który przygotowuje listy pasażerów. Pasażerowie mogą rezerwować bilety na różne loty w różnych biurach i przez internet. Wszystkie rezerwacje są zapisywane w biurze centralnym. Na podstawie pliku z rezerwacjami należy stworzyć pliki z listą pasażerów dla każdego lotu. Każda lista jest tworzona w odrębnym pliku. Nazwą pliku jest symbol lotu. W pliku umieszczona jest nazwa lotniska i data. W kolejnych liniach umieszczone są numery siedzeń i nazwiska pasażerów, posortowane według numerów.

2 Analiza zadania

Zagadnienie przedstawia problem wykonywania operacji na liście jednokierunkowej takich jak wczytanie z pliku i zapisanie do listy oraz zapis do pliku o nazwie jednego elementu z listy. Dodatkowym problemem było sprawdzenie poprawności danych zapisanych w pliku do wczytania.

2.1 Struktury danych

W programie wykorzystano struktury do przechowywania danych oraz realizacji listy jednokierunkowej. Wszystkie dane przechowywane były w zmiennych typu string, jednak w momencie zapisywania do pliku miejsce zarezerwowane dla pasażera było zmieniane na zmienną int. Dodatkowo użyta została zmienna typu bool, która określała obecność pliku o podanym kodzie lotu. Wykorzystana również została mapa, która ułatwiła zapisanie pasażerów posortowanych według numer miejsca.

2.2 Algorytmy

Najważniejszy algorytm dotyczy realizacji listy. Lista składa się ze składników oraz wskaźnika na kolejny element listy. Dla ostatniego elementu listy wskaźnik na następny element listy jest pusty (ma wartość nullptr). Nie jest przechowywany wskaźnik na poprzedni element listy, jest to więc lista jednokierunkowa. W ramach listy należało obsłużyć takie operacje jak wczytanie z pliku i zapis do listy, a następnie zapis tej listy do oddzielnych plików. Ważnym algorytmem również było sprawdzenie poprawności danych zarówno pod względem odpowiedniej ilości składników w każdej linijce jak i poprawności poszczególnych składników.

3 Specyfikacja zewnętrzna

3.1 Obsługa programu

Program jest uruchamiany z linii poleceń. Należy przekazać do programu nazwę pliku wejściowego z odpowiednim przełącznikiem:

```
...\nazwa_programu.exe -i nazwapliku.txt
```

Plik jest plikiem tekstowym i musi posiadać rozszerzenie .txt. Jeśli w folderze roboczym nie znajduje się plik o podanej nazwie zostanie wyświetlony komunikat:

```
Plik nie istnieje!
```

W przypadku braku poprawnego uruchomienia programu zostanie wyświetlony komunikat:

```
Niepoprawne wywołanie programu!
W celu poprawnego uruchomienia programu
należy użyć przełącznika "-i" oraz podać nazwę pliku z odlotami,
o rozszerzeniu .txt i znajdującym się w folderze roboczym.
W momencie poprawnego wywołania programu zwróci on wartość "0".
```

3.2 Format danych wejściowych

Dane dla poszczególnego lotu powinny być zapisane w jednej linijce, oddzielone od siebie spacją. Dane mają następującą kolejność: symbol lotu, lotnisko startowe, data lotu, nazwisko pasażera, numer miejsca. Symbol lotu składa się z 5 znaków, bez znaków specjalnych. Lotnisko startowe składa się tylko z liter, gdzie ich wielkość nie jest brana pod uwagę. Data lotu ma format RRRR-MM-DD. Nazwisko składa się tylko z liter, wielkość nie jest brana pod uwagę. Numer miejsca składa się tylko z cyfr.

3.3 Komunikaty

Komunikaty niezwiązane z obecnością pliku pojawiają się tylko w momencie niepoprawności danych. Zawierają one numer linijki, w której wystąpił błąd oraz czym spowodowany jest błąd. Przykładowe komunikaty:

```
w linijce 2 pliku test.txt kod lotu zawiera niedozwolone znaki.
Linijka została pominięta.
W linijce 4 pliku test.txt nazwisko zawiera niedozwolone znaki.
Linijka została pominięta.
W linijce 10 znajduja się niekompletne lub niepoprawne dane.
Linijka została pominięta.
```

4 Specyfikacja wewnętrzna

Program jest podzielony na kilka plików. W pliku:

- Marcin2.cpp - znajduje się funkcja, która sprawdza czy został użyty odpowiedni przełącznik. Jeśli tak to zostaje uruchomiona funkcja wczytująca, zapisująca i zwalniająca pamięć.
- funkcje.cpp - znajdują się funkcje, które odpowiedzialne są za zapisywanie, wczytywanie, tworzenie kolejnych elementów listy jednokierunkowej oraz zwalniające pamięć.
- sprawdzarka.cpp - znajdują się funkcje, które sprawdzają czy wczytywana linijka jest kompletna i zawiera poprawne dane.
- funkcje.h - znajduje się struktura odloty, która przechowuje dane o każdym locie oraz wskaźnik na kolejny lot. dodatkowo znajdują się tam nagłówki funkcji użytych w pliku: funkcje.cpp.
- sprawdzarka.h - znajdują się nagłówki funkcji użytywch w pliku: sprawdzarka.cpp

4.1 Zmienne zdefiniowane w programie

W programie zdefiniowano następujący typy danych:

- odloty - struktura przechowująca dane jednej linijki z pliku, zmienną informującą o istnieniu pliku oraz wskaźnik na kolejny element listy.

4.2 Ogólna struktura programu i szczegółowy opis implementacji funkcji

Główny plik programu to funkcje.cpp. Jego funkcje to:

```
1 odloty *stworz_odlot(string kod, string miasto, string
    data, string nazwisko, string miejsce);
2 void dodaj_odlot(odloty *&glowa, string kod, string
    miasto, string data, string nazwisko, string miejsce
    );
3 void wczytywanie(string nazwa, odloty *&glowa);
4 void zapisz_pasazerow(ofstream &plik, odloty *
    akt_odloty);
5 void zapisywanie(odloty *glowa);
6 void zwolnij(odloty *&glowa);
```

Omówienie poszczególnych składników pliku:

- odloty *stworz_odlot(**string** kod, **string** miasto, **string** data, **string** nazwisko, **string** miejsce) - funkcja która tworzy i zwraca wskaźnik na nowy lot z uzupełnionymi danymi, które podane są jako parametry.
- **void** dodaj_odlot(odloty *&glowa, **string** kod, **string** miasto, **string** data, **string** nazwisko, **string** miejsce) - funkcja, która dodaje kolejny lot do listy. Parametrami jest wskaźnik na pierwszy element oraz poszczególne dane lotu.
- **void** wczytywanie(**string** nazwa, odloty *&glowa) - funkcja, która wczytuje kolejne linijki pliku wejściowego oraz uruchamia funkcje odpowiedzialne za sprawdzenie poprawności danych oraz dodanie kolejnego lotu. Parametrami są nazwa pliku oraz wskaźnik na pierwszy element.
- **void** zapisz_pasazerow(ofstream &plik, odloty *akt_odloty) - funkcja, która zapisuje posortowanych według miejsca pasażerów do pliku. Wykorzystana jest tutaj mapa oraz jej iteracja. Parametrami są plik wejściowy oraz wskaźnik na pierwszy element listy.
- **void** zapisywanie(odloty *glowa) - funkcja, która tworzy plik o nazwie symbol_lotu.txt, a następnie wpisuje tam symbol lotu, lotnisko startowe oraz datę lotu. Następnie uruchamia funkcję odpowiedzialną za wpisanie pasażerów. Parametrem jest wskaźnik na pierwszy element listy.
- **void** zwolnij(odloty *&glowa) - funkcja, która usuwa loty z pamięci przed wyłączeniem programu. Zapobiega to wyciekom pamięci. Parametrem jest wskaźnik na pierwszy element listy.

Plik sprawdzarka.cpp zawiera funkcje:

```
1 bool podzial_na_stringi(int numer_linijki, string nazwa
    , ifstream &plik, string &kod, string &miasto,
    string &data, string &nazwisko, string &miejsce)
2 bool sprawdzenie_kodu(int linijka, string nazwa, string
     kod)
3 bool sprawdzenie_miasta(int linijka, string nazwa,
    string miasto)
4 bool sprawdzenie_daty(int linijka, string nazwa, string
     data)
5 bool sprawdzenie_nazwiska(int linijka, string nazwa,
    string nazwisko)
6 bool sprawdzenie_miejsca(int linijka, string nazwa,
    string miejsce)
7 bool sprawdzenie_poprawnosci_danych(int linijka, string
     nazwa, string kod, string miasto, string data,
    string nazwisko, string miejsce)
```

Omówienie poszczególnych składników struktury:

- **bool** podzial_na_stringi (**int** numer_linijki , **string** nazwa, ifstream &plik , **string** &kod, **string** &miasto, **string** &data, **string** &nazwisko, **string** &miejsce) - funkcja, która pobiera linijkę danych z pliku wejściowego, sprawdza poprawność zapisania danych, a następnie dzieli linijkę na poszczególne składniki. W momencie poprawnej linijki zwraca funckja zwraca wartość true. Parametrami są numer linijki, która jest pobierana, nazwa pliku, plik z danymi wejściowymi oraz zmienne, do których tymczasowo zapisywane są poszczególne dane lotu.

- **bool** sprawdzenie_kodu(**int** linijka , **string** nazwa, **string** kod) - funkcja, która sprawdza czy kod lotu jest poprawnie zapisany. Zwraca wartość true w przypadku poprawnego kodu lotu. Parametrami są numer linijki, nazwa pliku oraz kod lotu do sprawdzenia.

- **bool** sprawdzenie_miasta(**int** linijka , **string** nazwa, **string** miasto) - funkcja, która sprawdza czy miasto jest poprawnie zapisane. Zwraca wartość true w przypadku poprawnego miasta. Parametrami są numer linijki, nazwa pliku oraz miasto do sprawdzenia.

- **bool** sprawdzenie_daty(**int** linijka , **string** nazwa, **string** data) - funkcja, która sprawdza czy data jest poprawnie zapisana. Zwraca wartość true w przypadku poprawnej daty. Parametrami są numer linijki, nazwa pliku oraz data do sprawdzenia.

- **bool** sprawdzenie_nazwiska(**int** linijka , **string** nazwa, **string** nazwisko) - funkcja, która sprawdza czy nazwisko jest poprawnie zapisane. Zwraca wartość true w przypadku poprawnego nazwiska. Parametrami są numer linijki, nazwa pliku oraz nazwisko do sprawdzenia.
- **bool** sprawdzenie_miejsca(**int** linijka , **string** nazwa, **string** miejsce) - funkcja, która sprawdza czy miejsce jest poprawnie zapisane. Zwraca wartość true w przypadku poprawnego miejsca. Parametrami są numer linijki, nazwa pliku oraz numer miejsca do sprawdzenia.
- **bool** sprawdzenie_poprawnosci_danych(**int** linijka , **string** nazwa, **string** kod, **string** miasto, **string** data, **string** nazwisko, **string** miejsce) - funkcja, która uruchamia sprawdzenie poszczególnych danych lotu. Parametrami są numer linijki, nazwa pliku oraz poszczególne dane lotu, które zostają przekazane do funkcji sprawdzających poszczególne dane lotu.

Plik funkcje.h zawiera strukturę:

```
1 {
2     string kod;
3     string miasto;
4     string data;
5     string nazwisko;
6     string miejsce;
7     bool obecnosc_pliku;
8     odloty *nast pny;
9 };
```

Omówienie poszczególnych składników struktury:

- **string** kod - zmienna przechowująca kod lotu,
- **string** miasto - zmienna przechowująca miasto,
- **string** data - zmienna przechowująca datę lotu,
- **string** nazwisko - zmienna przechowująca nazwisko,
- **string** miejsce - zmienna przechowująca numer miejsca pasażera,
- **bool** obecnosc_pliku - zmienna przechowująca informację czy pasażer o danym kodzie lotu został już wpisany do pliku,
- odloty *nastepny - wskaźnik na kolejny element listy.

5 Testowanie

Program był testowany z plikiem, który zawierał zarówno poprawne dane jak i niepoprawne. Wszystkie niepoprawności zostały wyłapane i wypisane za pomocą odpowiednich komunikatów. Wszystkie dane poprawnie wpisane do pliku zostały zapisane do odpowiednio nowo powstałych plików.

6 Wnioski

Program do przygotowywania listy pasażerów nie jest programem bardzo skomplikowanym. Udało się zrealizować wszystkie cele projektu. Największe problemy sprawiło sprawdzenie poprawności danych oraz odpowiednie zapisanie ich. Ćwiczenia laboratoryjne były bardzo pomocne w pracy nad programem, szczególnie, że stosunkowo wcześnie został przerobiony materiał niezbędny do rozpoczęcia pracy nad programem. Projekt nauczył mnie pisania kodu czytelnego nie tylko dla mnie, a znacząco zwiększył umiejętność tworzenia funkcji, podziału programu na kilka plików, posługiwania się wskaźnikami oraz poznałem wiele nowych funkcji do sprawdzania danych. Ważnym aspektem było odpowiednie zarzadzanie pamiecia, aby uniknac wycieków pamieci. Zrealizowano to z wykorzystaniem destruktora struktur, które automatycznie kasuja niepotrzebne dane. Dodatkowym utrudnieniem był wymóg uzycia własnych implementacji list, bez korzystania z list w bibliotekach STL.