Politechnika Śląska
Wydział Informatyki, Elektroniki i Informatyki

Podstawy Programowania Komputerów

Autobusy

autor	Jakub Polczyk
prowadzący	dr inż. Marcin Połomski
rok akademicki	2019/2020
kierunek	informatyka
rodzaj studiów	SSI
semestr	1
termin laboratorium	poniedziałek, 08:30 – 10:00
sekcja	21
termin oddania sprawozdania	2020-01-22

1 Treść zadania

Proszę napisać program, który na podstawie planów jazdy autobusów generuje tabliczki przystankowe. Każda linia autobusowa ma plan jazdy autobusów zapisany w formie tekstowej. Linie autobusowe są oznaczone liczbami naturalnymi, dla każdego przystanku podana jest godzina w formacie `GG:MM`, a następnie podana jest jednowyrazowa nazwa przystanku, np:

```
LINIA 13
04:45     Rynek
04:53     Szkutnika
05:23     Reymonta
05:45     Dworzec

07:45     Rynek
07:53     Szkutnika
08:23     Reymonta
08:45     Dworzec

12:46     Rynek
12:56     Szkutnika
13:25     Reymonta
13:47     Dworzec
```

Wpisów w pliku może być dowolna liczba. Na podstawie plików wejściowych program generuje tabliczki przystankowe dla wszystkich przystanków w formacie:

```
RYNEK
=====

linia 6

03:15, 03:18, 03:23

linia 13

12:22, 12:55, 13:22, 13:55
```

Program uruchamiany jest z linii poleceń z wykorzystaniem następujących przełączników

`-i` pliki wejściowe z planami jazdy autobusów

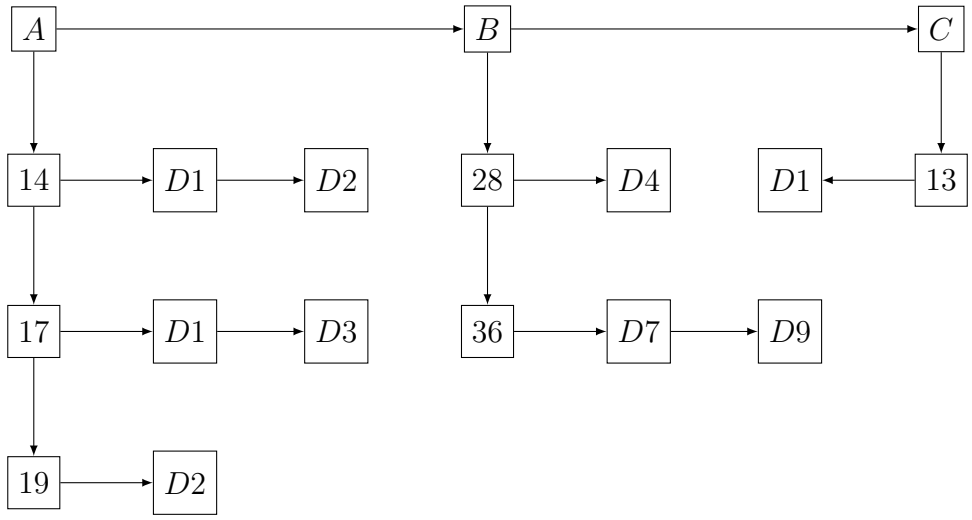
Tabliczki przystankowe zapisane są do plików. Każdy z nich ma nazwę tożsamą z nazwą przystanku.

2 Analiza zadania

Zagadnienie przedstawia problem utworzenia plików z przystankami generowanymi na podstawie linii autobusowych z plików wejściowych. [1]

2.1 Struktury danych

W programie wykorzystano listę w liście w liście. W Pierwszej liście przechowywane są `przystanki`. Każdy przystanek ma swoją `listę linii autobusowych` Każda linia autobusowa ma swoją `listę przystanków`. Taka struktura danych umożliwia łatwe uporządkowanie danych z pliku wejściowego.



Rysunek 1: Przykład zaimplementowanej struktury.

2.2 Algorytmy

Program wykorzystuje trzykrotne przeciążenie nazwy funkcji na każdym etapie wstawiając w odpowiednie miejsce do posortowanej rosnąco listy. Bardzo istotnym elementem tego algorytmu jest zwracanie wskaźnika na dany

element. Zwrócony adres umożliwa prawidłowe przypisane elementu do kolejnej listy. Wypisanie przystanków do plików wyjściowych wykorzystuje metody zarówno iteracyjne jak i rekurencyjne przechodzenia przez listę.

3 Specyfikacja zewnętrzna

Program jest uruchamiany z linii poleceń. Należy przekazać do programu obsługi nazwę pliku wejściowego po odpowiednim przełączniku (-i dla pliku wejściowego), np.

```
Bus -i LINIA1.txt
```

Program zabezpieczony jest przed nieprawidłowymi danymi wejściowymi. Niżej umieszczone będą możliwe otrzymane komunikaty. Podanie wyłącznie nazwy programu:

```
Blad skladniowy! Za malo elementow!
```

Błędna pozycja przełącznika `-i`:

```
Przelacznik -i powinien znajdowac sie od razu za nazwa programu!
```

Brak plików wejściowych:

```
Nie podano plikow wejsciowych!
```

Pliki wejściowe z rozszerzeniem innym niż .txt:

```
Zly format! Plik powinien byc z rozszerzeniem .txt!
```

Uniknięcie każdego z powyższych błędów umożliwi prawidłowe wykonanie programu i taki komunikat:

```
Poprawnie wprowadzono dane!
```

4 Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z paradygmatem strukturalnym. W programie rozdzielono interfejs (komunikację z użytkownikiem) od logiki aplikacji (tworzenie odpowiednio sformatowanych plików).

4.1 Ogólna struktura programu

W funkcji głównej wywoływana jest funkcja Przelacznik , która sprawdza poprawność wprowadzonych danych w konsoli. Błąd poskutkuje odpowiednim komunikatem. Kolejnym etapem jest wywołanie funkcji UploadLine, która pobiera zawartość pliku wejściowego i za pomocą funkcji Add dodaje w odpowiednie miejsce listy. Następnie wywoływana jest funkcja StopToFile , która dla każdego przystanku tworzy plik tekstowy i za pomocą funkcji ShowStop wypisuje dane ze strumienia. Na zakończenie wywoływana jest funkcja DeleteAll zwalniająca całą pamięć dynamicznie zaalokowaną w trakcie działania programu.

4.2 Szczegółowy opis typów i funkcji

Szczegółowy opis typów i funkcji zawarty jest w załączniku.

5 Testowanie

Program został przetestowany na różnego rodzaju plikach. Program został sprawdzony pod kątem wycieków pamięci.

6 Wnioski

Program Autobusy nie jest skomplikowanym programem, chociaż wymaga samodzielnego zarządzania pamięcią. Najbardziej wymagające okazało się sortowanie list, które z biegiem czasu doprowadzało do coraz fikuśniejszych, a zarazem prostszych w zarządzaniu struktur. Realizacja projektu nauczyła mnie jak dobrze posługiwać się wskaźnikami i jak operować na listach jednokierunkowych.

Literatura

[1] Krzysztof Simiński. Wykłady z podstaw programowania komputerów.

Dodatek
Szczegółowy opis typów i funkcji

Projekt zaliczeniowy z PPK-SSI

Wygenerowano przez Doxygen 1.8.17

Rozdział 1

Indeks klas

1.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

Rozdział 2

Indeks plików

2.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików z ich krótkimi opisami:

Rozdział 3

Dokumentacja klas

3.1 Dokumentacja struktury Line

`#include <struct.h>`

Atrybuty publiczne

- int LineIndex
  numer linii
- Time * pHeadTime
  wskaźnik na listę zawierająca godziny i minuty
- Line * pNextLine
  wskaźnik na kolejny element listy

3.1.1 Opis szczegółowy

Struktura opisująca linię

Dokumentacja dla tej struktury została wygenerowana z pliku:

- kod/struct.h

3.2 Dokumentacja struktury Stop

`#include <struct.h>`

Atrybuty publiczne

- std::string StopName
  nazwa przystanku
- Line * pHeadLine
  wskaźnik na listę linii
- Stop * pNextStop
  wskaźnik na następny przystanek

3.2.1 Opis szczegółowy

Struktura opisująca przystanek

Dokumentacja dla tej struktury została wygenerowana z pliku:

- kod/struct.h

3.3 Dokumentacja struktury Time

`#include <struct.h>`

Atrybuty publiczne

- int hour

  Godzina.
- int minute

  Minuta.
- Time * pNextTime

  Wskaźnik na kolejny element listy.

3.3.1 Opis szczegółowy

Struktura opisująca godzinę i minutę elementu linii

Dokumentacja dla tej struktury została wygenerowana z pliku:

- kod/struct.h

Rozdział 4

Dokumentacja plików

4.1 Dokumentacja pliku kod/fun.h

```
#include "struct.h"
```

Funkcje

- Stop ∗ Add (Stop ∗&pHead, const std::string &StopName)
- Line ∗ Add (Line ∗&pHead, int &LineIndex)
- Time ∗ Add (Time ∗&pHead, int hour, int minute)
- void ShowStop (std::ostream &stream, Stop ∗&pHead)
- void UploadLine (Stop ∗&pHeadStop, int argc, char ∗argv[ ])
- void StopToFile (Stop ∗&pHeadStop)
- void DeleteTime (Time ∗&pHead)
- void DeleteLine (Line ∗&pHead)
- void DeleteAll (Stop ∗&pHead)
- bool Przelacznik (int argc, char ∗argv[ ])

4.1.1 Dokumentacja funkcji

4.1.1.1 Add() [1/3]

```
Line* Add (
            Line *& pHead,
            int & LineIndex )
```

Funkcja dodaje linię do konkretnego przystanku

Parametry

pHead	Reprezentant listy linii
LineIndex	Numer linii

Zwraca

Wksaźnik na linię

4.1.1.2 Add() [2/3]

```
Stop* Add (
            Stop *& pHead,
            const std::string & StopName )
```

Funkcja dodaje przystanki do listy

Parametry

pHead	Reprezentant listy przystanków
StopName	Nazwa przystanku

Zwraca

Wksaźnik na przystanek

4.1.1.3 Add() [3/3]

```
Time* Add (
            Time *& pHead,
            int hour,
            int minute )
```

Funkcja dodaje godzinę oraz minutę do konkretnej linii

Parametry

pHead	Reprezentant listy godzin i minut
hour	Godzina
minute	Minuta

Zwraca

Wskaźnik na godziny i minuty

4.1.1.4 DeleteAll()

```
void DeleteAll (
            Stop *& pHead )
```

Funkcja usuwa wszystko

Parametry

pHead	Reprezentant listy przystanków

4.1.1.5 DeleteLine()

```
void DeleteLine (
            Line *& pHead )
```

Usuwa linię

Parametry

pHead	Reprezentant listy linii

4.1.1.6 DeleteTime()

```
void DeleteTime (
            Time *& pHead )
```

Funkcja usuwa godziny i minuty

Parametry

pHead	Reprezentant listy godzin i minut

4.1.1.7 Przelacznik()

```
bool Przelacznik (
            int argc,
            char * argv[] )
```

Funkcja sprawdza poprawność wprowadzonych danych w konsoli

Parametry

argc	Liczba elementów podana w konsoli
argv	Wskaźnik na tablicę znaków z konsoli

Zwraca

Funkcja zwraca wartość 1 jeżeli dane zostały wprowadzone poprawnie

4.1.1.8 ShowStop()

```
void ShowStop (
            std::ostream & stream,
            Stop *& pHead )
```

Funkcja wyświetla przystanek

Parametry

stream	Strumień wyjściowy
pHead	Reprezentant listy przystanków

4.1.1.9 StopToFile()

```
void StopToFile (
            Stop *& pHeadStop )
```

Funkcja zapisuje przystanki do plików tekstowych

Parametry

pHeadStop	Reprezentant listy przystanków

4.1.1.10 UploadLine()

```
void UploadLine (
            Stop *& pHeadStop,
            int argc,
            char * argv[] )
```

Funkcja wczytuje z wejścia linię

Parametry

pHeadStop	Reprezentant listy przystanków
argc	Liczba elementów podana w konsoli
argv	Wskaśnik na tablicê znaków z kosoli

4.2 Dokumentacja pliku kod/struct.h

Komponenty

- struct Time
- struct Line
- struct Stop

Indeks