Politechnika Śląska
Wydział Informatyki, Elektroniki i Informatyki

Podstawy Programowania Komputerów

Bank

autor	Maciej Sowiński
prowadzący	dr Ewa Lach
rok akademicki	2019/2020
kierunek	informatyka
rodzaj studiów	SSI
semestr	1
termin laboratorium	poniedziałek, 12:00 – 13:30
sekcja	61
termin oddania sprawozdania	2020-01-26

1 Treść zadania

Program do zarządzania bankiem. Na wejściu wczytujemy plik z kontami użytkowników (numer konta, dane osobowe użytkownika, limit debetu oraz zestawienie transakcji (data, kwota)). Program uruchamiany jest z linii poleceń z wykorzystaniem następujących przełączników:

- -fin - plik wejściowy
- -wy <nrkonta> - wypłata kwoty z konta
- -we <nrkonta> - wpłata kwoty na konto
- -wyciag - uzyskanie wyciągu z konta za określony okres (posortowane według daty) z podsumowaniem
- -undo <nrkonta><data><kwota> - cofnięcie transakcji (usunięcie transakcji)
- -wywe - wpłata dowolnej kwoty na konto innego użytkownika banku przez użytkownika banku
- -rT - zapisanie raportu wykonanych w banku operacji w danym okresie (posortowane według daty, kwot transakcji lub numerów kont)
- -rD - zapisanie raportu użytkowników, którzy mają debet na koncie (posortowane według nazwisk lub numerów kont).

2 Analiza zadania

Zagadnie przedstawia problem zarządzania bankiem. dane w pliku który zawiera konta bankowe użytkowników muszą być w odpowiednim formacie a miedzy kontami powinien znaleźć się ustalony i jednakowy separator.

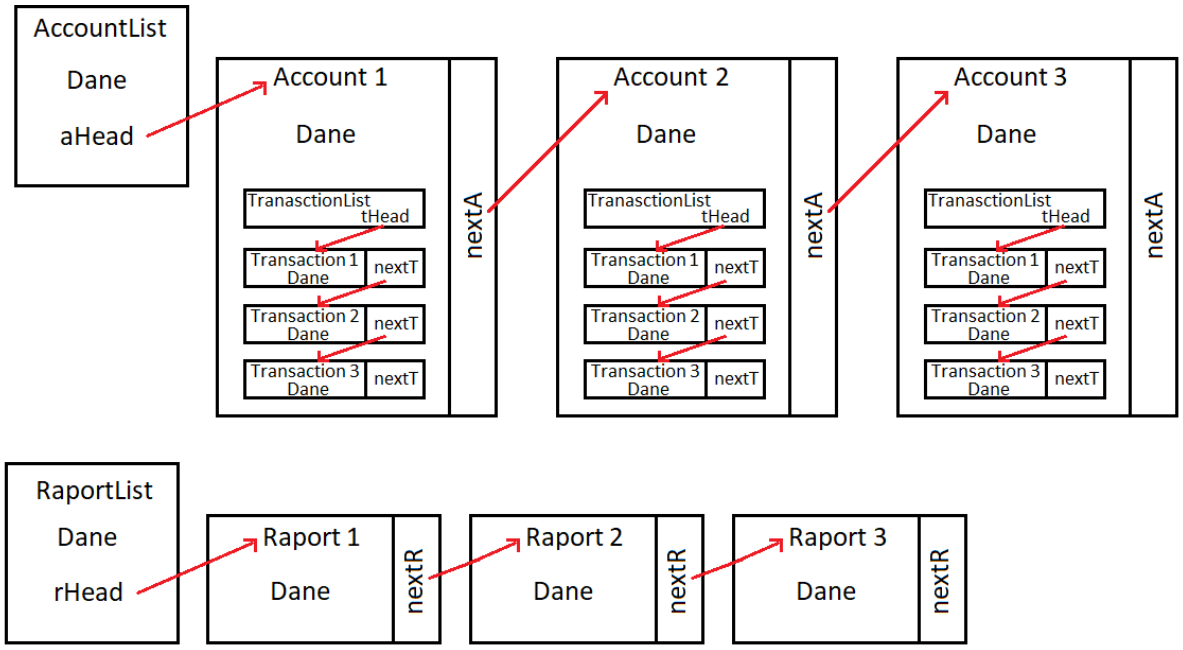
2.1 Struktury danych

W programie wykorzystano trzy listy jednokierunkowe. Lista przechowuje dane w swoich elementach. Element oprócz potrzebnych w zadaniu danych posiada wskaźnik na następny element. Lista kont dla każdego konta posiada oddzielną listę transakcji. Ostatnią listą jest lista raportu, która gromadzi na podstawie poprzednich list dane potrzebne do sporządzenia raportu. Powyższe struktury danych powodują ich logiczne i intuicyjne ułożenie co umożliwia wygodne korzystanie z treści w nich zawartych. Możliwość dodawania i usuwania wielu elementów dynamicznie, szukania, wyświetlania, porównywania, sortowania oraz tworzenia wyciągów z kont jak i całych raportów użytkowników czy transakcji. Rysunek 1 pokazuje wygląd struktur.

2.2 Algorytmy

Program wykonuje dodawanie elementu do listy umieszczając go na końcu. Znajduje i usuwa element o parametrach podanych przez użytkownika z listy (o ile istnieje) w dowolnym jego ustawieniu w liście (na początku, na końcu, w środku). Wyszukuje element o parametrach podanych przez użytkownika. Wyświetla jeden lub wiele elementów listy, a z konkretnego elementu może wyświetlić wszystkie lub tylko część danych w zależności od polecenia.

Porównuje elementy listy takie jak numer konta, nazwisko użytkownika, daty i kwoty transakcji. Sortowanie poszczególnych elementów odbywa się metodą bąbelkową o złożoności czasowej $O(n^2)$.
Wszystkie powyższe funckje operują na listach i przechodzą przez nie w sposób iteracyjny.



Rysunek 1 przedstawia wygląd użytych w programie struktur

3 Specyfikacja zewnętrzna

Program uruchamiany z linii poleceń. Należy przekazać do programu nazwe pliku wejściowego po przełączniku `-fin`. Następnie do odpowiednich przełączników odpowienie dane w pokazanej poniżej kolejności:

```
-wy <nr_konta> <kwota>
-we <nr_konta> <kwota>
-wyciag <nr_konta> <data_otwierająca_zakres>
<data_zamykająca_zakres>
-undo <nr_konta> <data> <kwota>
-wywe  <nr_konta_obciążonego> <nr_konta_odbiorcy> <kwota>
-rT <data_otwierająca_zakres> <data_zamykająca_zakres>
<typ_sortowania*>
-rD <typ_sortowania**>
```

* typ sortowania według: (1)daty, (2)kwot transakcji, (3)numerów kont, (inne)sortowanie nieaktywne
** typ sortowania według: (1)nazwisk, (2)numerów kont, (inne)sortowanie nieaktywne

Przykładowe poprawne wypisanie przełączników:

```
Bank -fin plik.txt -wy 12345678912345 530.12 -we
98745612378945 1002 -wyciag 12345678912345 2020-01-01 2020-10-
01
Bank -undo 12345678912345 2020-01-01 530.12 -fin lista.txt -
wywe 98745612378945 12345678912345 123.34 -rT 2020-01-01 2020-
10-01 2
-rD 1
```

Pliki są plikami tekstowymi. Przełączniki mogą być podane w dowolnej kolejności. Uruchomienie programu z parametrem -h np.:

```
Bank -h
Bank -fin plik.txt –h -wy 12345678912345 530.12
```

powoduje wyświetlenie krótkiej pomocy.

Podanie nieprawidłowej nazwy pliku powoduje wyświetlenie komunikatu:

```
Nie znaleziono pliku <nazwa_pliku.txt>
```

4 Specyfikacja wewnętrzna

W programie rozdzielono interfejs (komunikację z użytkownikiem) od logiki aplikacji (operacji wykonywanych na kontach i transakcjach).

4.1 Ogólna struktura programu

W funkcji głównej znajduję się pętle obsługujące przełączniki. Pętla zewnętrzna bada obecność przełączników `-fin -h`. Pętla wewnętrzna w której znajdują się pozostałe przełączniki wykona się tylko wtedy gdy wśród podanych przełączników znajduje się przełącznik `-fin` w przeciwnym przypadku zostaje wypisany stosowny komunikat i program kończy się. Wywołana jest funkcja **readData**. Wczytywane są dane z pliku i odpowiednio umieszczane w listach, jeśli plik nie istnieje zwracany jest stosowny komunikat i program kończy się. Następnie w zależności od podanych parametrów wykonywane są odpowiednie funkcję:

`-wy` wywoływana jest funkcja **findAccount** która zwraca wskaźnik na użytkownika lub **NULL** kiedy nie istnieje, następnie funkcja **withdrawal**, która zaokrągla kwotę funkcją **roundAmount** sprawdza jej poprawność i dostępne środki na koncie, pobiera aktualną datę z funkcji **getCurrentDate**, a następnie wywołuję funkcję **addTransaction** dodającą transakcję do listy transakcji.
`-we` wywoływana jest funkcja **findAccount** która zwraca wskaźnik na użytkownika lub **NULL** kiedy nie istnieje, następnie funkcja **deposit**, która zaokrągla kwotę funkcją **roundAmount**

sprawdza jej poprawność, pobiera aktualną datę z funkcji **getCurrentDate**, a następnie wywołuję funkcję **addTransaction** dodającą transakcję do listy transakcji.

-wyciag wywoływana jest funkcja **findAccount** która zwraca wskaźnik na użytkownika lub **NULL** kiedy nie istnieje, następnie funkcja **accountStatement** która sprawdza poprawność zakresu dat przez funkcje **checkDatesOrder**. Transakcję są sortowane funkcją **dateSort**, a następnie sprawdzane czy mieszczą się w zakresie przez funkcję **checkDateInRange** i wyświetlane. Na końcu wyświetlane jest podsumowanie.

-undo wywoływana jest funkcja **findAccount,** która zwraca wskaźnik na użytkownika lub **NULL** kiedy nie istnieje, następnie wywoływana jest funkcja **findTransaction** która zwraca wskaźnik na transakcję lub **NULL** kiedy nie istnieje, na końcu wywołana funkcja **deleteTranasaction** która usuwa transakcję.

-wywe wywoływana jest dwa razy funkcja **findAccount** która zwraca wskaźnik na użytkownika lub **NULL** kiedy nie istnieje, następnie funkcja transfer, która zaokrągla kwotę funkcją **roundAmount** sprawdza jej poprawność i dostępne środki na koncie, pobiera aktualną datę z funkcji **getCurrentDate**, a następnie dwa razy wywołuję funkcję **addTransaction** dodającą transakcję do listy transakcji.

-rT wywoływana jest funkcja **raportTransaction**, która sprawdza poprawność zakresu dat przez funkcje **checkDatesOrder**. Następnie sprawdzane jest czy transakcje mieszczą się w zakresie przez funkcję **checkDateInRange** i dodawane do listy raportu funkcją **addItem**. Lista jest sortowana funkcją **sortRaportTransaction** a następnie wyświetlana przez funkcję **showRaportTransaction** pamięć zostaje zwolniona przez polecenie **delete.**

-rD wywoływana jest funkcja **raportDebitUsers**, która sprawdza środki użytkownika a następnie dodaje do listy raportu funkcją **addItem**. Lista jest sortowana funkcją **sortRaportDebit** a następnie wyświetlana przez funkcję **showRaportDebit** pamięć zostaje zwolniona przez usunięcie listy raportu polecenie **delete**.

Program kończy się wywołaniem funkcji **saveData**, która zapisuje wszystkie dane z list do podanego na początku pliku. Potem następuje zwolnienie pamięci po przez polecenie **delete**, które usuwa listy.

4.2 Szczegółowy opis typów i funkcji

Szczegółowy opis typów i funkcji zawarty jest w załączniku.

5 Testowanie

Program został przetestowany na kilku plikach. Brak pliku powoduje zgłoszenie błędu. Plik pusty nie powoduje zgłoszenia błędu, ale utworzenie pustego pliku wynikowego. Program został sprawdzony pod kątem wycieków pamięci.

6 Wnioski

Dzięki dobremu rozplanowaniu struktur zadanie nie było bardzo skomplikowane jednak dosyć czasochłonne. Odpowiednia ilość tematycznie utworzonych plików .h i .cpp znacznie ułatwiła dopisywanie nowych funkcji i korzystanie z już istniejących, a także szukania najprostszego rozwiązania problemów stawianych przez kolejne podpunkty zadania. Projekt pomógł zrozumieć jak działa lista oraz pokazał jak można na niej operować. Program jest otwarty na dalszy rozwój o dodatkowe struktury czy funkcje. Największym problemem były drobne błędy które ciężko było znaleźć, a lista błędów nie pomagała w ich szukaniu.