Politechnika Śląska w Gliwicach
Wydział Automatyki, Elektroniki i Informatyki

Książki

autor	Tomasz Migała
prowadzący	mgr inż. Marek Kokot
rok akademicki	2017/2018
kierunek	teleinformatyka
rodzaj studiów	SSI
semestr	1
termin laboratorium	piątek, 11:30 - 12:00
grupa	2
sekcja	3
termin oddania sprawozdania	25-01-18
data oddania sprawozdania	25-01-18

1. Treść zadania

Napisać program przyporządkowujący autorów i ich teksty do określonych sekcji (imitacja biblioteki) z wykorzystaniem listy, oraz drzewa binarnego. Autorzy, ich teksty i etykiety zawarte są w pliku tekstowym w formacie : Autor; Tytuł; Etykieta, Etykieta. Przyporządkowani autorzy zostaną zapisani do pliku według Etykiet posortowani alfabetycznie. Program uruchamiany jest z linii poleceń z wykorzystaniem następujących przełączników:

-i plik wejściowy z danymi do posortowania
-o plik wyjściowy z posortowanymi danymi

2. Analiza zadania

Zagadnienie przedstawia problem przyporządkowywania odpowiednich danych do odpowiednich działów pobranych z pliku, a następnie posortowanie i zapisanie do innego pliku.

2.1 Struktury danych

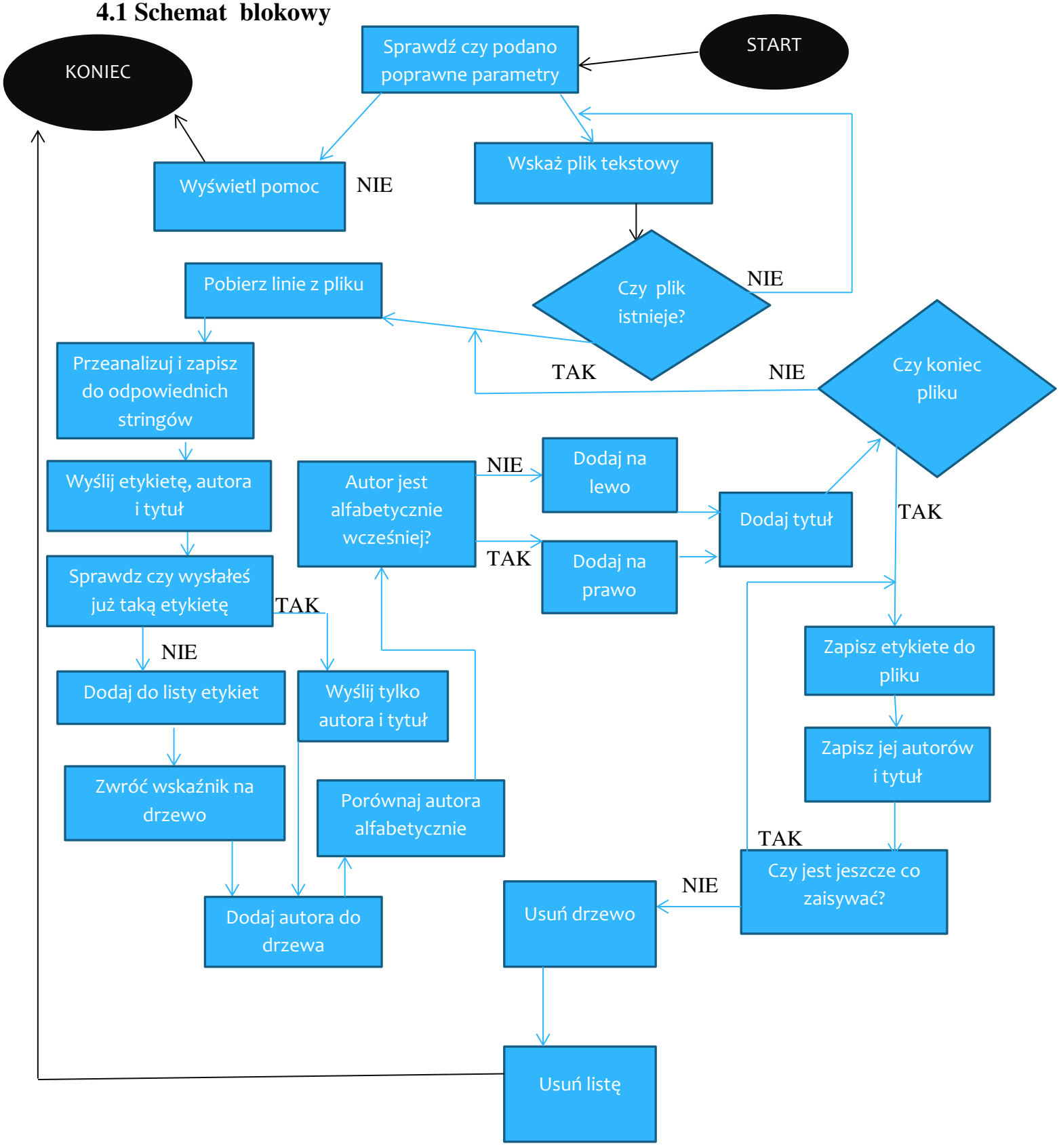
W programie użyto listy której każdy element to Etykieta jak i wskaźnik na początek drzewa binarnego. Drzewo binarne przechowuje w swoich węzłach Autorów i Tytuły, najpierw wpisywani do drzewa są autorzy alfabetycznie od lewej, potem do każdego autora przypisywane są jego dzieła.

3. Specyfikacja zewnętrzna

Program jest uruchamiany z linii poleceń. Należy przekazać do programu nazwy plików: wejściowego i wyjściowego po odpowiednich przełącznikach (odpowiednio: -i dla pliku wejściowego i -o dla pliku wyjściowego), np.
projekt -i Książki.txt -o posortowane.txt
projekt -o posortowane.txt - i Książki.tx

4. Specyfikacja wewnętrzna

4.1 Schemat blokowy



Pliki są plikami tekstowymi. Przełączniki mogą być podane w dowolnej kolejności. Uruchomienie programu bez parametrów, lub z parametrem -h spowoduje wyświetlenie krótkiej pomocy. Uruchomienie programu z nieprawidłowymi parametrami powoduje wyświetlenie komunikatu:

```
----------------------------------------------------------
Prawdopodobnie podałeś błędne parametry!!
----------------------------------------------------------
Program: Książki | Wersja: 1.00
Twórca: Tomasz Migała
----------------------------------------------------------
Program pobiera dane z pliku, przetwarza je w odpowiedni sposób,
a następnie sortuje według nazwisk autorów, przyporządkowanych,
do odpowiednich działów." << "\n";
----------------------------------------------------------
Poprawne parametry: -i Ksiazki.txt -o output.txt
----------------------------------------------------------
```

Podanie nieprawidłowej nazwy pliku powoduje wyświetlenie odpowiedniego komunikatu:

```
----------------------------------------------------------
Nie znaleziono takiego pliku.
Poprawna nazwa pliku: Ksiazki.txt
----------------------------------------------------------
```

Jeżeli w pliku wejściowym coś będzie nie tak spowoduje to wyświetlenie odpowiedniego komunikatu:

```
----------------------------------------------------------
Sprawdz czy w pliku wejsciosym 'Ksiazki.txt'
wszystkie dane wprowadzone sa
wedlug formatu
AUTOR; TYTUL; ETYKIETA, ETYKIETA1 (opcjonalnie)
----------------------------------------------------------
Poprawne parametry: -i Ksiazki.txt -o output.txt
----------------------------------------------------------
```

Program został z paradygmatem strukturalnym. W programie rozdzielono interfejs(komunikację z użytkownikiem) od logiki aplikacji (przypisywanie autorów do działów).

4.1 Struktury zdefiniowane w programie:

```
struct Drzewo
{
        String   autor;
        string   tytul;

        Drzewo* lewy;
        Drzewo* prawy;
};
```

```
struct Lista
{
        string etykieta;
        Drzewo* korzen;
        Lista* nast;
        Lista* poprz;
};
```

```
struct Parametry
{
        string wejscie;
        string wyjscie;

        bool wejscie_podane = false;
        bool wyjscie_podane = false;
        bool pomoc_podane = false;
};
```

4.2 Opis funkcji

1. **bool Sprawdz_Parametry(int argc, char *argv[], Parametry &parametry)**
   Sprawdza czy podane parametry są zgodne z założonymi, jeżeli nie, wyświetla pomoc “pokaz_Pomoc”.

2. **void Pokaz_Pomoc()**
   Jeżeli funkcja Sprawdz_Parametry zwóci fałsz wyświetla pomoc.

3. **void Cos_Nie_Tak(int i)**
   Jeżeli w programie któraś część będzie nie poprawna wyświetla odpowiedni komunikat i kończy dalszą pracę programu.

4. **bool Pobierz_Z_Pliku(string plik, Lista* &glowa, Lista* &ogon)**
   Funkcja sprawdza czy plik istnieje. Jeżeli nie ma takiego pliku zwraca false i wyświetla stosowny komunikat, w przeciwnym wypadku, do zmiennej typu string zapisuje linie, po czym szuka w niej “;”, “,”, zapisuje słowa do przeznaczonych do tego zmiennych typu string (Autor, Tytuł, Etykieta) .

4. **bool Sprawdz_Czy_Etykieta_Istnieje(Lista* glowa, string &etykieta)**
   Funkcja ta porównuje kolejne elementy listy z wysłaną do niej etykietą dopóki istnieje głowa. Jeżeli element listy jest taki sam zwraca True.

5. **bool Na_Poczatek(Lista* &glowa, Lista* &ogon, Drzewo* korzen, string &etykieta)**
   Wykorzystje funkcję Sprawdz_Czy_Etykieta_Istnieje, jeżeli funkcja ta zwróci True, zwraca False, w przeciwnym wypadku kontynuuje działanie i dopisuje podaną jej etykietę na początek listy.

7. **Drzewo* Zwroc_Wskaznik_na_drzewo(Lista* glowa, string &etykieta)**
   Dopóki istnieje głowa porównuje elementy listy z dostarczonym do niej stringiem, jeżeli etykiety są sobie ‘równe’ zwraca wskaźnik na drzewo.

8. **Drzewo* Dodaj_Do_Drzewa(Drzewo* &korzen, string autor, string &tytul)**
   Na początek sprawdza, czy korzeń nie jest nullem, jeżeli nie sprawdza, czy poprzedni dodany autor jest pod względem alfabetowym wcześniej czy później jeżeli później to teraźniejszy dodawany autor umieszczany jest z lewej strony, jeżeli nie to z prawej.

9. **void Zapisz_Autorow(fstream &file, Drzewo* korzen)**
   Dopóki istnieje korzeń, zapisuje autorów, oraz tytuły do pliku w formacie Autor; Tytuł.

10. **bool Rozpocznij_Zapis(Lista* glowa, string &nazwa_pliku)**
    Tworzy plik do zapisu, sprawdza czy udało się go utworzy, jeżeli nie zwraca false, jeżeli tak Zapisuje Etykiety, a następnie korzysta z fuckji **Zapisz_Autorow.**

11. **void Usun_Drzewo(Drzewo* &korzen)**
    Rekurencyjnie przechodzi całe drzewo, a nastęnie je usuwa.

12. **void Usun_Liste(Lista* &glowa)**
    Usuwa nie potrzebną już listę

5. Wnioski

Zadanie zostało zrealizowane pomyślnie. Program nie wydaje się być skomplikowany w swojej budowie, jednakże sprawiał on mi wiele problemów, jako początkującemu w dziedzinie programistycznej. Największą trudnością było utworzenie drzew jako kolejne elementy Listy Etykiet. Podczas tworzenia tego programu mniej więcej udało mi się zrozumieć sens korzystania z list jak i drzew binarnych.