Politechnika Śląska w Gliwicach

Wydział Informatyki, Elektroniki i Informatyki

**Podstawy Programowania**

**Komputerów**

Książki

|  |  |
| --- | --- |
| autor | Mateusz Wilczyński |
| prowadzący | dr inż. Karolina Nurzyńska |
| rok akademicki | 2014/2015 |
| kierunek | informatyka |
| rodzaj studiów | SSI |
| semestr | 2 |
| termin laboratorium / ćwiczeń | Czw. 08:30 – 10:00 |
| grupa | 4 |
| termin oddania sprawozdania | 2015-06-11 |
| data oddania sprawozdania | 2015-06-11 |
|  |  |

# Treść zadania

Książki w bibliotece opisane są przez nazwisko autora, tytuł i etykiety. Etykiety przyporządkowują książki do działów. Każda książka może mieć kilka etykiet. Zadanie polega na zapisaniu do pliku książek wg etykiet. Książki opatrzone tą samą etykietą są posortowane wg nazwiska autora.

# Analiza zadania

Zagadnienie przedstawia problem wczytania z pliku danych, a następnie zapisanie ich do innego pliku w zmienionym formacie.

## Struktury danych

Etykiety są przechowywane w sortowanej liście jednokierunkowej. Nazwiska autorów oraz tytuły są zapisane w liście podwieszanej pod etykietami. Nazwisko i tytuł są przechowywane w jednej strukturze danych. Przedstawiona struktura danych jest strukturą dynamiczną umożliwiającą dodawanie różnej ilości danych.

## Algorytmy

Algorytm służący do wczytywania z pliku wczytuje autora, tytuł oraz listę etykiet. Po wczytaniu następuje sprawdzenie, czy dane są poprawne, oraz czy format linii jest odpowiedni. Jeśli jakieś kryterium nie jest spełnione, to dalsze linie nie są wczytywane. Jeśli wczytanie autora, tytuł oraz listy etykiet przebiegło pomyślnie następuje dodanie tych wartości do dynamicznej struktury danych. Lista etykiet jest dzielona na pojedyncze etykiety, dla każdej etykiety następuje dodanie jej do listy etykiet, oraz stworzenie pod nią listy podwieszanej zawierającej odpowiedniego autora oraz tytuł. Jeśli dana etykieta istnieje już w liście, to autor oraz tytuł są dodane pod istniejącą etykietę. Zarówno etykiety jak i autor oraz tytuł są dodawane do odpowiednich list alfabetycznie. Lista podwieszana jest dodawana alfabetycznie na podstawie nazwiska autora, w przypadku gdy jest ono takie same dla kilku książek, to są one ułożone na podstawie tytułu. Po dodaniu do listy wszystkich etykiet z danej linii algorytm przechodzi do następnej.

Algorytm zapisujący wynik działania do pliku zaczyna od głowy listy (*head),*  która zostaje przypisana do lokalnej zmiennej wskaźnikowej *wsk*. Na początku następuje zapisanie pola *label*  struktury na którą wskazuje wskaźnik do pliku, następnie odbywa się zapis listy podwieszonej do danego wskaźnika. Rozpoczyna się ono od elementu *wsk->under*  jest to pierwszy element listy podwieszanej, następuje zapis pola *line* tego elementu do pliku, następnie przechodzimy do elementu *next* i znowu zapisujemy jego pole *line*  do pliku. Powtarzamy to dopóki element *next* jest różny od *NULL,* wtedy kończy się zapis listy podwieszonej pod daną etykietę i następuje przejście do następnej etykiety poprzez *wsk->next*. Wyżej opisane działania są powtarzane dopóki *wsk->next* jest różny od *NULL.*

# Specyfikacja zeewnętrzna

Program można uruchomić przez linię poleceń wpisując *książki.exe –i input.txt –o output.txt*. Polecenie to trzeba wpisać znajdując się w folderze zawierającym plik *książki.exe* Jeśli nie zostaną podane oba parametry programu, zostanie wyświetlony komunikat: *Niewlasciwe parametry programu!* a następnie program zakończy działanie.

Podczas uruchamiania programu po przełączniku *–i ­*podajemy nazwę pliku wejściowego. Plik ten zawiera dane na których będzie działał program. Jeśli plik ma rozszerzenie inne niż *.txt* zostanie wyświetlony komunikat *Plik [nazwa] ma niewlasciwe rozszerzenie!* a następnie program zakończy działanie. Jeśli podany plik nie istnieje zostanie wyświetlony komunikat *Nie udalo sie otworzyc pliku [nazwa!* a następnie program zakończy działanie. Przykładowa zawartość pliku:

*Platon; Uczta; filozofia  
Shakespeare; Hamlet; tragedia, barok*

Po przełączniku *–o* podajemy nazwę pliku wyjściowego, do którego zostanie zapisany wynik działania programu. . Jeśli plik ma rozszerzenie inne niż *.txt* zostanie wyświetlony komunikat *Plik [nazwa] ma niewlasciwe rozszerzenie!* a następnie program zakończy działanie. Jeśli podany plik nie istnieje zostanie wyświetlony komunikat *Nie udalo sie zapisac do pliku [nazwa!* a następnie program zakończy działanie.

Jeśli nie wystąpiły wyżej opisane błędy program zapisze swój rezultat do pliku wyjściowego, wyświetli komunikat *Dane zostaly zapisane do pliku [nazwa],* a następnie zakończy swoje działanie.

# Specyfikacja wewnętrzna

## Zmienne funkcji main

|  |  |
| --- | --- |
| \*input | Wskaźnik na typ **FILE**, wskazuje na plik wejściowy |
| \*head | Wskaźnik na typ **list\_el,** przechowujemy w niej adres na początku listy. |
| \* input\_file\_name | Wskaźnik na typ **char,** służy do przechowywania nazwy pliku wejściowego. |
| \*output\_file\_name | Wskaźnik na typ **char,** służy do przechowywania nazwy pliku wejściowego. |

## Funkcje i Procedury

|  |
| --- |
| **int get\_files\_names(int argc, char \*\* argv, char \*\*input\_file\_name, char \*\*output\_file\_name);** |
| Funkcja służy do wczytywania parametrów programu do odpowiednich zmiennych.  Parametry:   * **\*\*input\_file\_name** –adres wskaźnika na ciąg znaków, który ma przechowywać plik wejściowy, * **\*\* output \_file\_name** –adres wskaźnika na ciąg znaków, który ma przechowywać plik wyjściowy.   Funkcja zwraca *0* jeśli udało się odczytać oba parametry lub *-1* w przeciwnym wypadku. |

|  |
| --- |
| **int read\_file(FILE\* plik, list\_el\*\* head);** |
| Fukcja służy do wczytywania danych z pliku, oraz przy użyciu innych procedur wstawienia tych danych do listy.  Parametry:   * **\* plik** – zmienna służąca do przechowywania adresu na plik, * **\*\*head** – Wskażnik na adres początku listy.   Funkcja zwraca *0*  jeśli wykonanie przebiegło pomyślnie, lub *-1*  dane w pliku mają niewłaściwy format. |

|  |
| --- |
| **void add\_to\_list(list\_el\*\* head, char \*word, char \*author, char \*title);** |
| Procedura dodaje lub wyszukuje element listy, oraz przy użyciu innych procedur dodaje element do listy podwieszanej.  Parametry to :   * **\*\*head** – wskaźnik na adres początku listy, * **\*word –**zmienna przechowująca słowo, które ma zostać wstawione jako etykieta do listy, * **\* author –**zmienna przechowująca autora, który ma zostać wstawiony do listy podwieszanej, * **\* title –**zmienna przechowująca tytuł, który ma zostać wstawiony do listy podwieszanej, |

|  |
| --- |
| **void add\_to\_u\_list(list\_el\* label\_head, char \*author, char \*title);** |
| Procedura dodaje element do listy podwieszanej.  Parametry to :   * **\* label\_head** – adres, elementu głównej listy pod który ma zostać wstawiony nowy element, * **\* author –**zmienna przechowująca autora, który ma zostać wstawiony do listy podwieszanej, * **\* title –**zmienna przechowująca tytuł, który ma zostać wstawiony do listy podwieszanej, |

|  |
| --- |
| **void clean\_memory(list\_el\*\* head);** |
| Procedura służy do zwalniania pamięci zarezerwowanej na listę oraz przy użyciu innych procedur pamięci przydzielonej na listy podwieszone pod każdy z elementów.  Parametry:   * **\*\*head** – wskaźnik na adres początku listy, zajmującej pamięć, która ma zostać zwolniona. |

|  |
| --- |
| **void clean\_u\_memory(list\_el\* label\_head);** |
| Procedura służy do zwalniania pamięci zarezerwowanej na listę podwieszoną pod element podany jako parametr  Parametry:   * **\* label\_head** – wskaźnik na element, pod który jest podwieszona lista zajmująca pamięć, która ma zostać zwolniona. |

|  |
| --- |
| **void save\_list(char \*file\_name, list\_el \*head);** |
| Procedura służy do zapisywania zawartości listy głównej oraz przy użyciu innych procedur zawartości list podwieszonych do pliku.  Parametry:   * **\*file\_name** – nazwa pliku do któtego ma zostać zapisana treść listy. * **\*head** – wskaźnik na adres początku listy, której zawartość ma zostać zapisana. |

|  |
| --- |
| **void save\_u\_list(FILE\* output, u\_list\_el \*u\_head);** |
| Procedura służy do zapisywania zawartości listy podwieszanej do pliku.  Parametry:   * **\*file\_name** – nazwa pliku do któtego ma zostać zapisana treść listy. * **\* u\_head** – element, pod który jest podwieszona lista, która ma zostać zapisana do pliku. |

|  |
| --- |
| **void strip(char \*str);** |
| Procedura usuwa puste znaki z początku i końca elementu podanego jako parametr  Parametry:   * **\*str** – wskaźnik na ciąg znaków na którym ma zostać wykonana powyższa operacja. |

|  |
| --- |
| **int check\_filename\_ext(char \*filename);** |
| Funkcja sprawdza, czy podana nazwa pliku posiada rozszerzenie *.txt*  Parametry:   * **\*filename** – wskaźnik na ciąg znaków zawierający nazwę pliku, która ma zostać sprawdzona.   Funkcja zwraca *0*  jeśli nazwa pliku posiada rozszerzenie *.txt* lub *-1*  w przeciwnym wypadku. |

# Testowanie

Program został przetestowany na poprawnych plikach .txt oraz na plikach z innymi rozszerzeniami. Dla niepoprawnych oraz nieistniejących plików program wyświetla informację o błędzie.

Testowanie:

* Zawartość pliku we.txt:

*Platon; Uczta; filozofia*

*Shakespeare; Hamlet; tragedia, barok*

*Cormen; Algorytmy; informatyka, podrecznik*

*Mickiewicz; Pan Tadeusz; klasyka, romantyzm, lektura*

*Akwinata; Summa Teologiczna; filozofia, sredniowieczeAkwinata; Dumma Teologiczna; filozofia, sredniowieczeAkwinata; xyz Teologiczna; filozofia, sredniowiecze*

Parametry programu:

-i we.txt -o wy.txt

Został wyświetlony komunikat: *Dane zostaly zapisane do pliku wy.txt*

Zawartość pliku wy.txt:

*barok*

*Shakespeare; Hamlet*

*filozofia*

*Akwinata; Dumma Teologiczna*

*Akwinata; Summa Teologiczna*

*Akwinata; xyz Teologiczna*

*Platon; Uczta*

*informatyka*

*Cormen; Algorytmy*

*klasyka*

*Mickiewicz; Pan Tadeusz*

*lektura*

*Mickiewicz; Pan Tadeusz*

*podrecznik*

*Cormen; Algorytmy*

*romantyzm*

*Mickiewicz; Pan Tadeusz*

*sredniowiecze*

*Akwinata; Dumma Teologiczna*

*Akwinata; Summa Teologiczna*

*Akwinata; xyz Teologiczna*

*tragedia*

*Shakespeare; Hamlet*

# Wnioski

Program książki nie jest programem skomplikowanym. Program składa się z kilku ważnych części: listy podwieszanej tworzonej na podstawie odpowiedniego pliku wejściowego podanego w parametrach programu oraz zapis tej listy do pliku wyjściowego również podanego w parametrach programu. Największą trudność stanowiło odpowiednio dodawanie danych do listy oraz sprawdzanie poprawności pliku wejściowego. Program został podzielony na wiele funkcji i procedur, zapewnia to dużą izolację kodu, co jest korzystne przy większych programach, zwiększa to też łatwość późniejszej zmiany kodu.