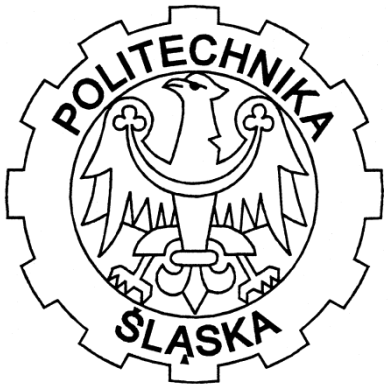
Politechnika Śląska w Gliwicach

Wydział Automatyki, Elektroniki i Informatyki



PODSTAWY PROGRAMOWANIA KOMPUTERÓW

„BIBLIOGRAFIA”

Autor	Aleksandra Kaleta
Prowadzący	mgr inż. Marek Kokot
Rok akademicki	2015/2016
Kierunek	Teleinformatyka
Rodzaj studiów	SSI
Semestr	1
Termin laboratorium	poniedziałek
Grupa	1
Sekcja	2
Termin oddania sprawozdania	29.01.2016.
Data oddania sprawozdania	31.01.2016.

1 Treść zadania

W publikacjach często stosuje się odnośniki do literatury. Ręczne numerowanie jest dość uciążliwe, ale można do tego zaprząc komputer, który to zrobił szybko, bez marudzenia i bez błędów.

W pliku z tekstem znajdują się odnośniki do literatury zbudowane zgodnie ze schematem:

\cite{nazwa etykiety}

Przykładowy plik:

W dziele **\cite{sh-hamlet}** autor przedstawia sytuacje Danii **\cite{dania2011}**. ...

Opisy bibliograficzne, zawarte w osobnym pliku, maja następujaca postać:

<etykieta>

<imie i nazwisko autora>

<tytuł>

Zakładamy, ze imię i nazwisko autora to dwa słowa rozdzielone spacjami. Etykieta może być zbudowana z liter, cyfr, myślnika, podkreślnika i dwukropka. Poszczególne opisy w pliku oddzielone są pustą linia.

Przykładowy plik:

sh-hamlet

William Shakespeare

Hamlet

dania2011

Jan Florjanowicz

Rocznik Statystyczny 2011

grahmat

Robert Graham

Matematyka konkretna

W wyniku działania programu zostanie utworzony plik wynikowy, który na końcu będzie miał dodany akapit o nazwie Bibliografia, w którym będą znajdowały się opisy bibliograficzne dzieł cytowanych w pliku z tekstem artykułu. Te książki, które nie są cytowane, ale występują w pliku z opisami bibliograficznymi, nie są umieszczane w bibliografii. Książki w bibliografii są posortowane wg nazwisk autorów. Książki w bibliografii są ponumerowane. W tekście wynikowym nie pojawiają się znaczniki **\cite{...}**, ale odpowiednie numery w nawiasach kwadratowych.

Przykład pliku wynikowego:

W dziele [2] autor przedstawia sytuacje Danii [1].

...

Bibliografia

[1] Jan Florjanowicz, Rocznik Statystyczny 2011

[2] William Shakespeare, Hamlet

Program uruchamiany jest z linii poleceń z wykorzystaniem następujących przełączników (kolejność przełączników jest dowolna):

-t plik wejściowy z tekstem

-o plik wyjściowy

-b plik z opisami bibliograficznymi

2 Analiza zadania

Celem projektu jest przećwiczenie implementacji i korzystania z dynamicznych struktur danych (np. listy, drzewa) i zarzadzania pamięcią w programie. Warunkiem sine qua non jest użycie w programie tych struktur. Użycie bibliotecznych kontenerów (np. vector, list) nie spełnia tego warunku.

2.1 Struktury danych

W programie wykorzystano drzewo binarne do przechowywania etykiet. Drzewo binarne przechowuje dane w węzłach, które mogą mieć od 0 do 2 potomków. Napisy w etykietach będą porównywane alfabetycznie. Na lewo idą elementy zaczynające się na literę alfabetu, która jest wcześniej niż elementy idące na stronę prawą.

Drugą strukturą jest lista dwukierunkowa. Wykorzystuję ją, aby posortować nazwiska autorów alfabetycznie, które później dodam do bibliografii. W liście nadaje numery odpowiednim etykietom, które będą wyświetlane w publikacji.

2.2 Algorytmy

W drzewie umieszczane są etykiety pobierane z pliku. Tworzę wskaźnik w funkcji z listą dwukierunkową, aby pobrać etykiety z drzewa i je posortować alfabetycznie. W funkcji z listą porównuję etykiety za pomocą funkcji warunkowej, aby program je odpowiednio posortował.

3 Specyfikacja zewnętrzna

Program jest uruchamiany z linii poleceń. Należy przekazać do programy nazwy plików: plik wejściowy z tekstem, plik wyjściowy oraz plik z opisami bibliograficznymi:

-t plik wejściowy z tekstem

-o plik wyjściowy

-b plik z opisami bibliograficznymi

4 Specyfikacja wewnętrzna

4.1 Typy zdefiniowane w programie

W programie zdefiniowano następujące typy:

```
struct etykiety_ksiazek
{
    string etykiety;
    string imie;
    string nazwisko;
    string tytul;
    etykiety_ksiazek* lewa;
    etykiety_ksiazek* prawa;

};
```

Ten typ struktury służy do tworzenia drzewa binarnego. Zdefiniowano również wskaźnik do przekazywania informacji na lewy i prawy węzeł.

```
struct Lista_dwukierunkowa
{
    Lista_dwukierunkowa* nastepny;
    Lista_dwukierunkowa* poprzedni;
    etykiety_ksiazek* etykiety;
    int numer;
};
```

Ten typ struktury służy do tworzenia listy dwukierunkowej. Dodano wskaźniki na elementy następne i poprzednie. Wprowadzono wskaźnik na **etykiety,** aby przekazać je z drzewa binarnego do listy. Zdefiniowano zmienną **numer,** aby później dodawać kolejne numery odpowiadające etykietom w liście.

4.2 Ogólna struktura programu

W funkcji głównej wywołana jest funkcja:

```
bool parse_parameters(int argc, char**argv, string&
input_file_name, string& input_file_name_2, string&
output_file_name)
```

Funkcja ta sprawdza, czy program został wywołany w prawidłowy sposób. Wtedy funkcja zwraca true lub false. Wykorzystywane są do tego parametry wymienione w pkt. 3 (Specyfikacja zewnętrzna). Po sprawdzeniu parametrów w zmiennych plikowych przechowywane są nazwy pliku wejściowego i wyjściowego. Następnie wywoływana jest funkcja.

```
void Czytaj_Opisy(const string& input_file_name_2,
etykiety_ksiazek*& root)
```

Funkcja ta otwiera plik z etykietami (input_file_name_2) i funkcją **getline** pobiera napisy z odpowiednich etykiet: imię, nazwisko oraz tytuł.

Bufor sczytuje pustą linię pomiędzy jednym tytułem, a drugim.

```
void Czytaj_Publikacje(const string& input_file_name,
etykiety_ksiazek*& root, Lista_dwukierunkowa*& head)
```

Funkcja otwiera plik z publikacjami(input_file_name.

W tej funkcji szukamy frazy „\cite{}”, aby zastąpić ją nawiasami kwadratowymi([]), by móc dalej przeszukiwać plik. W nawiasy zostaje później wpisana odpowiednia cyfra.

```
void Nadaj_numer(Lista_dwukierunkowa*& head)
```

Ta funkcja nadaje numery kolejnym elementom listy dwukierunkowej.

```
void Zastap_numerem(Lista_dwukierunkowa*& head, const
string& input_file_name, const string&
output_file_name)
```

Tutaj wykorzystano dwa pliki: etykiety(**input_file_name_2**) oraz Publikacje(**input_file_name**). W tej funkcji fraza „\cite{etykieta}” zostaje zastąpiona odpowiadającym numerem w liście. **Bufor** sczytuje pustą linię pomiędzy jednym tytułem, a drugim.

```
void Wypisz_bibliografie(Lista_dwukierunkowa*& head,
const string& output_file_name)
```

W tej funkcji wyświetlam już gotowe publikacje z bibliografią. Funkcja `ios::app` służy do dopisywania kolejnych elementów w pliku wyjściowym `output_file_name`.

```
void usun_drzewo(etykiety_ksiazek*& root)
```

Ta funkcja usuwa drzewo. Po kolei usuwane są węzły, a na koniec korzeń i przypisywana mu jest wartość null.

```
void usun_liste(Lista_dwukierunkowa*& head)
```

Ta funkcja usuwa listę dwukierunkową. Dodano wskaźnik, który przesuwa się po elementach listy na elementy następne, usuwając je.

5. Testowanie

Program wyświetla komunikat z błędem, kiedy zostają podane niepoprawne parametry. Kiedy podamy np. trzy imiona autora, program nie pobierze dwóch imion, tylko jedno, do białego znaku. Plik wyjściowy jest plikiem pustym, program sam go uzupełnia.

Program został sprawdzony pod kątem wycieków pamięci.

6. Wnioski

Program do tworzenia opisów bibliograficznych jest wg mnie dość ciekawy. Tworzenie go było całkiem trudnym zadaniem. Największym problemem było stworzenie posortowanej listy dwukierunkowej. Nie obyło się bez problemów również przy innych funkcjach, ale ostatecznie program działa.

Bardzo przydatne były notatki z laboratoriów dotyczące operacji na drzewach binarnych oraz listach. Niestety nie znalazłam w materiałach z wykładów, czy ćwiczeń potrzebnych informacji na temat mojego problemu z listą dwukierunkową. Myślę, że program działa dość sprawnie i znajdujące się tam informacje posłużą komuś za inspirację do lektury.