Politechnika Śląska w Gliwicach Wydział Automatyki, Elektroniki i Informatyki

**Podstawy Programowania Komputerów**

System kontroli wersji

Autor: Wojciech Bębenek

Prowadzący: J. Nalepa

Rok akademicki: 2017/2018

Kierunek: Informatyka

Rodzaj studiów: SSI

Semestr: 1

Termin laboratoriów/ćwiczeń: Czwartek 13:00-14:30

Grupa:5

Sekcja: 14

Termin oddania sprawozdania: 2018-01-28

Data oddania sprawozdania: 2018-01-25

1.Treść zadania

Napisać program służący do prostej kontroli wersji. Program uruchamiany jest z wiersza poleceń. Umożliwia on wyszukanie różnic pomiędzy dwoma różnymi plikami, nadpisanie różnic nową zawartością, przeglądanie logów, które zapisywane są dla konkretnego pliku w nowym pliku tekstowym z przyrostkiem ‘\_log’ w nazwie.

2.Analiza zadania

Zagadnienie przedstawia problem wyszukiwania różnic, oraz zapisywania ich do plików.

2.1 Struktury danych

W programie wykorzystano listę jednokierunkową. Użyta jest ona jako tablica dynamiczna przechowująca numery wierszy zawierających zmiany oraz ich zawartości. Użycie struktury było konieczne, aby nie ograniczać programu do konkretnego rozmiaru tablicy. Jednokierunkowość listy nie jest w tym przypadku minusem, gdyż program nie wymaga wyszukiwania, a same dane wyświetlane są po kolei, od góry do, dołu.

2.2 Algorytmy

Program pobiera od użytkownika nazwy plików, oraz wybrany przełącznik operacji. Wyszukiwanie różnic odbywa się na zasadzie pobierania wierszy z plików tekstowych oraz przyrównywaniu ich do siebie. Jeśli jakieś linie różnią się od siebie, tworzony jest nowy element listy, w którym zapisywany jest numer wiersza, w którym znaleziono różnicę, zawartość wiersza z pierwszego oraz drugiego pliku i wskaźnik do kolejnego elementu. Następnie różnice te są wyświetlane, bądź zapisywane do jednego, lub drugiego pliku, w zależności od wybranej operacji. Tworzony jest również plik z logami, który zapisuje zmiany, jakie wystąpiły w danym pliku.

3.Specyfikacja zewnętrzna

Program jest uruchamiany z linii poleceń. Do programu należy przekazać odpowiedni przełącznik, nazwę pierwszego pliku wejściowego, oraz drugiego pliku wejściowego. Nie należy wpisywać rozszerzeń pluku, np.

program.exe -d file1 file2

program.exe -c file1 file2

Pliki powinny być plikami tekstowymi z rozszerzeniem '.txt' oraz znajdować się w tym samym folderze co program. przełączniki powinny być podane w napisanej kolejności. Uruchomienie programu z parametrem -h, np.

program.exe -h

powoduje wyświetlenie pomocy. Podanie nieprawidłowej nazwy chociażby jednego pliku spowoduje wyświetlenie komunikatu

File does not exist

program.exe –l file1

Podanie parametru –l oraz nazwy pliku spowoduje wyświetlenie się wszystkich zmian jakie nastąpiły w tym pliku przy pomocy tego programu.

4.Specyfikacja wewnętrzna

4.1 Ogólna struktura programu

W funkcji głównej wywołana jest pętla sprawdzająca parametry uruchomienia programu, mająca na celu rozróżnianie parametrów oraz ich ilości. Wywołane są również funkcje sprawdzające istnienie pliku oraz poprawność nazewnictwa pliku, gdyż niektóre znaki są zakazane w nazwach. Są to funkcje:

W funkcji głównej wywołana jest pętla sprawdzająca parametry uruchomienia programu, ma ona na celu rozróżnianie parametrów oraz ich ilości. Wywołana jest również funkcja sprawdzająca czy dany plik w ogóle istnieje, jest to:

int CheckFileName(std::string filename\_1, std::string filename\_2)

Przed wykonaniem tej funkcji sprawdzane jest również, czy użytkownik nie wybrał czasem opcji wyświetlenia pomocy lub logu. Ta druga ma własną funkcję sprawdzającą dla jednego ciągu znaków.

Po sprawdzeniu poprawności przechowywane są nazwy plików oraz numer identyfikacyjny operacji, następnie wywoływana jest jedna z następujących funkcji:

void DisplayHelp();

void DisplayLog(std::string filename\_1);

void Diff(std::string filename\_1, std::string filename\_2);

void Commit(std::string filename\_1, std::string filename\_2);

void Update(std::string filename\_1, std::string filename\_2);

Funkcja pierwsza wyświetla pomoc. Funkcja druga otwiera plik z logami i wyświetla go w konsoli. Każda następna funkcja otwiera dwa pliki, sczytuje dane i zapisuje różnice w liście jednokierunkowej i w zależności od operacji są wykonywane różne działania. Po zakończeniu funkcji program zamyka pliki i kończy działanie.

4.2 Szczegółowy opis implementacji funkcji

void Diff(std::string filename\_1, std::string filename\_2);

void Commit(std::string filename\_1, std::string filename\_2);

void Update(std::string filename\_1, std::string filename\_2);

Do funkcji tej przekazywane są nazwy dwóch plików. Program tworzy obiekty klasy fstream file1 oraz file2. Następnie otwiera je z parametrem Ios::In. Tworzone są zmienne klasy string Templine1, Templine2 oraz klasy int counter1, counter2 oraz anotherone. Następnie uruchamiane są pętle while, które przy użyciu zmiennych counter1 oraz counter2 sprawdzają ile poszczególne pliki mają linii. Pliki następnie są zamykane i otwierane z tym samym parametrem. Następnie tworzony jest nowy obiekt struktury wskazujący na null. Następnie w zależności od tego, który plik ma więcej linii uruchamiana jest pętla sczytująca dane z dwóch plików. Jeśli czytany plik pierwszy jest dłuższy od drugiego, to puste linijki przyjmują również ‘pustą’ wartość. Jeśli wiersze różnią się od siebie, uruchamiana jest funkcja:

void add\_end(int NumberOfLine, std::string Content,std::string ContentM, Differences\*&head)

Przekazywane są do niej numer wiersza, zawartość wierszy z poszczególnych plików, oraz wskaźnik. Ma ona na celu utworzenie nowego elementu listy, na końcu, aby przy wyświetlaniu linie były wyświetlane po kolei.

Następnie wartość anotherone oraz line jest inkrementowana, następnie pliki są zamykane.

**Diff**

Ta część jest tylko dla funkcji Diff.   
Uruchamiana jest pętla while, która za zadanie ma wyświetlić wszystkie różniące od siebie się wiersze wraz z numerami linii. Następnie lista jest usuwana przy pomocy funkcji remove\_list.

**Commit & Update**

Ta część dotyczy funkcji Commit oraz funkcji Update.  
Tworzony jest nowy obiekt klasy fstream log\_file, dla którego przypisujemy nazwę pliku, w którym zachodziły jakieś zmiany z przyrostkiem \_log.txt, następnie plik ten jest otwierany z parametrami wejściowymi Ios::out oraz Ios::app. Następnie przy pomocy pętli while wszystkie elementy listy są zapisywane do pliku, jeśli plik już istniał, dane są dopisywane. Plik z logami jest zamykany, a cała lista usuwana wspomnianą wcześniej funkcją remove\_list.

Dla funkcji Commit pliki są otwierane ponownie, z parametrami wejściowymi Ios::In dla file1 oraz Ios::In, Ios::out, Ios::trunc dla file2. Następnie przy pomocy funkcji while sczytywane są wiersze z pliku pierwszego i na bieżąco zapisywane do wyczyszczonego pliku file2. Dla funkcji update jest dosłownie odwrotnie. Dla file1 dostajemy parametry Ios::In, Ios::out oraz Ios::trunc, a dla file2 samo Ios::In. Dane z file2 zapisywane są do file1 tą samą metodą.

void remove\_list(Differences \*&head)

Funkcja remove\_list pobiera tylko wskaźnik głowy, a następnie przy pomocy wskaźnika helper oraz pętli while usuwane są kolejno wszystkie elementy listy.

int CheckFileName(std::string filename\_1, std::string filename\_2)

Do funkcji przekazywane są nazwy dwóch plików wejściowych, które następnie otwierane są z parametrami tylko do odczytu. Jeśli nie udało się otworzyć któregoś z nich, to zwracana jest wartość 0, jeśli otworzone poprawnie zostaną dwa pliki, funkcja zwraca wartość równą 1.

void DisplayHelp()

Funkcja czyści okno konsoli oraz wyświetla informacje dotyczące prawidłowego korzystania z programu.

void DisplayLog(std::string filename\_1)

Funkcja ma za zadanie utworzenie nowego obiektu klasy fstream i otworzenie pliku z logami. Jeśli nie istnieje to program kończy działanie wraz z wyświetleniem komunikatu ‘File does not exist!’. W innym wypadku okno jest czyszczone, a zawartość pliku jest wypisywana przy pomocy pętli while w oknie konsoli. Następnie plik jest zamykany i program kończy działanie

5.Testowanie

Program był testowany na plikach tekstowych z rozszerzeniem ‘.txt’. Wszystkie pliki były poprawnie otwierane, a znalezione różnice w liniach były prawidłowe. Jeśli do jeden plik w danym miejscu ma pusty wers, to za jego linię nie jest nic wyświetlane. Zapisując pustą linię do drugiego pliku otrzymujemy również pustą linię. Najdłuższym plikiem, dla którego sprawdzono poprawność programu miał powyżej 100 linii, dla dłuższych nie został on sprawdzony.

6.Wnioski

Program do prostej konsoli wersji plików nie należał do najtrudniejszych, ale pomógł mi zrozumieć w lepszym stopniu działanie wskaźników. I wskaźniki również były największym wyzwaniem, jak również sama lista. Jednak po utworzeniu algorytmu zapisującego dane do listy wszystko poszło z górki i projekt został bez większych przeszkód doprowadzony do końca.