Politechnika Śląska w Gliwicach Wydział Automatyki, Elektroniki i Informatyki

**Podstawy Programowania Komputerów**

Szyfrowanie metodą podstawieniową

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: 2017-12-01

Data oddania sprawozdania: 2017-12-01

1.Treść zadania

Napisać program szyfrujący i rozszyfrowujący pliki metodą podstawieniową. Program uruchamiany jest z pliku wykonywalnego. Zaszyfrowany lub rozszyfrowany plik zostanie zapisany w tym samym folderze z dodanym przedrostkiem e (dla plików zaszyfrowanych) lub de (dla plików odszyfrowanych).

2.Analiza zadania

Zagadnienie przedstawia problem zaszyfrowywania i odszyfrowywania plików tekstowych.

2.1 Struktury danych

W programie wykorzystano ciągi znaków używanych jako tablice. Umożliwia to przetwarzanie pojedynczych znaków

2.2 Algorytmy

Program pobiera nazwę pliku oraz przesunięcie znaków od użytkownika. Szyfrowanie odbywa się przez pobieranie wierszy pliku tekstowego i zamiany kolejnych znaków w wierszu na znaki o n (przesunięcie podane przez użytkownika) większych wartościach według tablicy znaków ASCII. Rozszyfrowywanie odbywa się w sposób podobny, jednak znaki zamieniane są na znaki o wartościach o n mniejszych.

3.Specyfikacja zewnętrzna

Program jest uruchamiany z linii poleceń. Do programu należy przekazać odpowiedni przełącznik, nazwę pliku wejściowego(bez rozszerzenia), oraz poziom przesunięcia (liczba całkowita), np.

program.exe -e niezakodowane 5

program.exe -d zakodowane 5

Pliki powinny być plikami tekstowymi z rozszerzeniem '.txt'. przełączniki powinny być podane w napisanej kolejności. Uruchomienie programu z parametrem -h, np.

program.exe -h

powoduje wyświetlenie pomocy. Uruchomienie programu z niewłaściwymi parametrami skutkuje wyświetleniem komunikatu

Invalid parameters

Podanie nieprawidłowej nazwy pliku spowoduje wyświetlenie komunikatu

File does not exist

bądź poinformuje nas o nieprawidłowych znakach w nazwie pliku.

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:

void CheckFilename(std::string filename)

int CheckIfExist(std::string filename)

Po sprawdzeniu poprawności przechowywane są nazwa pliku, numer identyfikacyjny operacji oraz poziom przesunięcia. Następnie wywoływana jest funkcja:

void Encoding(std::string filename, int displacement)

Lub funkcja:

void Decoding(std::string filename, int displacement)

Funkcje otwierają plik, sczytują kolejne linie i umieszcza je na bieżąco w pliku wyjściowym. Następnie wykonywane jest, zależnie od wybranej operacji, kodowanie lub odkodowanie pliku. Po zapisaniu wszystkich linii program zamyka plik i kończy działanie.

4.2 Szczegółowy opis implementacji funkcji

void Decoding(std::string filename, int displacement)

void Encoding(std::string filename, int displacement)

Do funkcji tej przekazywane są nazwa pliku oraz poziom przesunięcia. Program tworzy obiekty klasy fstream **FileToProcess** oraz DecodedFile. Następnie przypisuje plik o nazwie **filename** do **FileToProcess** isprawdza czy dany plik istnieje, jeśli nie, to kończy działanie. Jeśli plik istnieje, program sprawdza, czy istnieje plik o nazwie wyjściowej, jeśli istnieje, sprawdza czy istnieje plik z numerem 2 w nawiasie, jeśli istnieje, sprawdza dalej, a potem tworzy plik z pierwszym dostępnym numerem w nawiasie. Plik dostępny jest pod nazwą **DecodedFile** jest on otwierany. Po utworzeniu i otworzeniu pliku użyta jest pomocnicza zmienna **TempLine**, do której zapisywane będą kolejne linie pliku. W pętli while zadeklarowana jest zmienna **DecodedLine**, która czyszczona będzie co jeden obrót pętli. Następnie, wciąż w pętli, utworzona jest kolejna pętla, tym razem for, która za zadanie ma sczytywanie kolejnych znaków, od indeksu 0, do indeksu o jeden mniejszego niż rozmiar tablicy. Utworzona została pomocnicza zmienna **actualchar** przechowująca znak, który kodowany jest na daną iterację pętli for, do zmiennej **echar** przypisywany jest znak z **actualchar**, którego kod ascii jest zmieniony o +5 (-5, w przypadku dekodowania). Następnie do zmiennej **DecodedLine** dopisywany jest znak z **echar**. Po zakończeniu pętli for wartość zmiennej **DecodedLine** dopisywana jest do pliku **DecodedFile** wraz z zakończeniem linii. Następnie obydwa pliki są zamykane.

void CheckFilename(std::string filename)

Do funkcji przekazana jest nazwa pliku pod nazwą filename. Funkcja tworzy zmienną **error**, do której przypisana zostaje wartość zero. Następnie utworzona jest pętla for, która jw. Sczytuje kolejne znaki i przypisuje je do pomocniczej zmiennej **actualchar.**Następnie do zmiennej **v** przypisana jest wartość ascii znaku z **actualchar**. Następnie instrukcja warunkowa sprawdza czy wartość **v** nie jest równa jednej z zakazanych wartości podanych w pliku, które nie mogą wystąpić w nazwie pliku w systemie windows. Jeśli takowy znak sięznajdzie funkcja zostaje przerwana, a zmiennej **error** jest przypisana wartość 1. Funkcja przechodzi do kolejnych instrukcji warunkowych. Jeśli **error**  jest równe 0, to funkcja przechodzi po prostu do nowej linii. W innym przypadku czyści ekran i wyświetla komunikat "Invalid characters in file name", następnie funkcja kończy działanie programu.

int CheckIfExist(std::string filename)

Do funkcji przekazana jest nazwa pliku pod nazwą **filename**. Funkcja tworzy obiekt klasy fstream **FileToProcess.** Następnie próbuje otworzyć ten plik z zamiarem odczytu. Następnie sprawdzany jest funkcją .good(), która przy wartości false zwraca wartość 0, a w innym wypadku 1.   
Wartość 0, już poza funkcją później skutkuje wyczyszczeniem konsoli oraz wyświetleniem komunikatu informującym o tym, że plik nie istnieje, wartość 1 pozwoli wykonywać się programowi dalej.

5.Testowanie

Program był testowany na plikach tekstowych z rozszerzeniem '.txt'. Wszystkie pliki były poprawnie szyfrowane (jeśli oczywiście przeszły instrukcje warunkowe związane z argumentami), to samo tyczy się odszyfrowywania. Jeśli przetwarzany plik jest pusty, tworzy pusty plik wynikowy z rozszerzeniem '.txt'. Największy plik, dla którego został uruchomiony program, to plik tekstowy o rozmiarze 247MB, dla większych nie został on sprawdzony.

6.Wnioski

Program do kodowania i dekodowania plików metodą podstawieniową było dosyć proste, aczkolwiek nauczyło mnie paru nowych trików. Wymogi pozwoliły mi również na zachowanie większego porządku w kodzie źródłowym niż dotychczas. Po raz pierwszy program stworzony przeze mnie pobierał parametry bezpośrednio z linii poleceń, aby nie wchodził on w interakcję z użytkownikiem. Największym wyzwaniem było doprowadzenie kodu do względnego porządku.