Program szyfrujący w języku C – dokumentacja

Horobets Kiril

Ilya Rohoutseu

# Cel projektu

Celem tego projektu jest stworzenie oprogramowania jako kompresora/dekompresora plików opartego na algorytmie Huffmana, wykorzystującego kodowanie binarne. Algorytm Huffmana jest bezstratną statystyczną metodą kompresji danych, która każdemu bajtowi ciągu wejściowego przypisuje unikalny kod o stałej lub zmiennej długości. W tym przypadku znaki występujące najczęściej otrzymują krótszy kod, natomiast znaki występujące rzadko otrzymują kod dłuższy.

# Dane wejściowe

Przychodzące dane będą listą znaków, wszystkich znaków w pliku.

# Argumenty wywołania programu

# Użytkownik może zbudować program za pomocą polecenia „make”.

Użytkownik może uruchomić program za pomocą polecenia „/.kompresor „ , podając plik źródłowy oraz plik do zapisu.

Jeżeli nie zostaną podane żadne argumenty, zostanie wyświetlony komunikat o błędzie.

Przykłady wywołania programu:

./kompresor –i input.txt -o output.txt

**# Makefile**

Makefile służy do budowania i czyszczenia programu, który wykonuje kodowanie Huffmana na plikach.

Pierwsza linia określa regułę, że cel "kompresor" zależy od plików "main.o" i "huffman.o", które mają być używane podczas kompilacji i linkowania.

Druga i trzecia linia określają reguły kompilacji kodu źródłowego "main.c" do pliku obiektowego "main.o" oraz kompilacji kodu źródłowego "huffman.c" do pliku obiektowego "huffman.o".

Ostatnia linia definiuje regułę czyszczenia, która ma usunąć pliki "kompresor", "main.o" i "huffman.o" utworzone podczas procesu budowania. Aby wykonać tę operację, wystarczy wywołać polecenie "make clean".

kompresor: main.o huffman.o

$(CC) -o kompresor main.o huffman.o

main.o : main.c

$(CC) -c main.c -o main.o

huffman.o : huffman.c

$(CC) -c huffman.c -o huffman.o

clean:

rm kompresor \*.o

**Dane wyjściowe**

# Wynikiem działania programu jest zaszyfrowane wyrażenie, zapisane również w postaci bitowej i zawarte w pliku code.txt.

# Pierwsza liczba w pliku to liczba pełnych bajtów, druga to liczba pojedynczych bitów, a następnie znaki i ich kody

# Przykład działania programu:

Przykład pliku wejściowego input.txt : aaaabbbbccccddddeeeffg

# Przykładowy plik wyjściowy output.bin : 11011011 01101111 11111111 00000000 01010101 10010010 01011101 11010000

Przykład pliku wyjściowego code.txt :

7 5

a 110

b 111

c 00

d 01

e 100

f 1011

g 1010

# Nasz kod implementuje algorytm Huffmana do kompresji plików tekstowych. Oto co robią funkcje i metody w tym kodzie:

# read\_file(FILE\* file) - odczytuje znaki z pliku i przechowuje numer każdego znaku w tablicy liczników.

# huffman\_code() - wykonuje algorytm Huffmana i tworzy tablicę kodów dla każdego znaku.

# search\_min() - wyszukuje najmniejszą wartość z liczników i zwraca ją.

# search\_tab(int min, int ind) - szuka w tablicy strchar elementu o najmniejszej wartości, który nie jest równy min i ma indeks inny niż ind. Zwraca indeks znalezionego elementu.

# make\_code(char\* first, char\* second) - dodaje do kodu tablicy 0 do zakodowania znaku pierwszego i 1 do zakodowania znaku drugiego.

# remove\_tab(int fi, int si) - łączy strchar[fi].str i strchar[si].str, dodaje strchar[si].prob do strchar[fi].prob i zwraca strchar[si].prob równe zero.

# code\_reverse() - odwraca każdy kod w tablicy kodów.

# print\_code(FILE\* plik) - wypisuje tablicę kodów do pliku.

# write\_text(FILE\* file, FILE\* in) - zapisuje zakodowany tekst do pliku.

# write\_bits\_to\_file(FILE\* file, const char\* binary\_string) - zapisuje zakodowane bity do pliku.

# Teoria

Kod Huffmana to algorytm kompresji danych, który przekształca źródłowy zestaw znaków w ciąg bitów o zmiennej długości. Opiera się on na założeniu, że często występujące znaki powinny zajmować mniej miejsca niż znaki występujące rzadziej.

Algorytm Huffmana rozpoczyna się od utworzenia tablicy częstotliwości dla wszystkich znaków w danych wejściowych. Następnie tworzony jest las drzew, gdzie każdy znak jest reprezentowany jako własne drzewo. Drzewa są łączone parami, zaczynając od najrzadziej występujących znaków, aż powstanie jedno drzewo.

Po połączeniu drzew tworzony jest nowy węzeł, który ma wagę równą sumie wag dwóch węzłów liści, które zostały połączone. Ten nowy węzeł zajmuje miejsce węzła macierzystego. Wybierany jest węzeł o mniejszej wadze dla lewego węzła dziecka oraz węzeł o większej wadze dla prawego węzła dziecka.

Następnie dla każdego symbolu tworzona jest ścieżka od korzenia drzewa do węzła odpowiadającego temu symbolowi. Jeśli symbol występuje częściej niż inne, jego kod będzie krótszy, a jeśli występuje rzadziej, jego kod będzie dłuższy. Kod symbolu jest reprezentowany jako ciąg bitów, gdzie każda ścieżka od korzenia do liścia drzewa jest określona przez bit 0 lub 1. Kod znaku składa się z sekwencji tych bitów.

Po utworzeniu tablicy kodów dla wszystkich znaków wykonywana jest kompresja danych. Dla każdego znaku jego kod źródłowy jest zastępowany kodem Huffmana. Dzięki temu zmniejsza się ilość danych i szybciej przesyłane są informacje. Odwrotny proces działa podczas dekodowania: ciąg bitów jest zamieniany na znaki na podstawie tablicy kodów Huffmana.

Kod Huffmana jest szeroko stosowany w różnych dziedzinach, takich jak kompresja audio i wideo, kompresja plików i przesyłanie danych przez sieci.

# Komunikaty błędów

Program stara się w każdy sposób pomóc użytkownikowi, na przykład :

* W przypadku, gdy użytkownik nie wpisał lub wpisał nieprawidłową wartość w wyborze wejścia, program wypisze podpowiedź typu "usage: ./kompresor -i <plik wejściowy> -o <plik wyjściowy>" .
* Program poinformuje również użytkownika o ewentualnym błędzie podczas otwierania pliku «Plik nie zostal otwarty»