Teoria informacji i kodowania - ćwiczenia laboratoryjne
Zadanie 2

Napisać program w języku C++ tworzący na podstawie modelu źródła informacji wyznaczonego wcześniej dla dowolnego pliku drzewo kodowania i tabelę kodową dla tego pliku.

W tym celu należy z pliku wejściowego wczytać do odpowiednio utworzonej tablicy tabelę symboli (bajtów 8-bitowych) posortowaną według liczby wystąpień i korzystając z metody Huffmana wyznaczyć i zapisać w pliku wynikowym tabelę opisującą drzewo kodowania, a następnie wyznaczyć na podstawie drzewa kodowania i zapisać w pliku wynikowym tabelę kodową.

Szczegółowe wymagania stałe:

- program ma się poprawnie kompilować w środowisku Dev-C++ w wersji dostępnej w sali laboratoryjnej, w której prowadzone są zajęcia;
- cały program ma mieścić się w jednej jednostce kompilacyjnej (jednym pliku *.cpp);
- program ma być uruchamiany z okna konsoli tekstowej, bez interfejsu graficznego;
- program ma rozpocząć i zakończyć swoje działanie bez potrzeby dodatkowych działań ze strony użytkownika (poza jego uruchomieniem); w szczególności bez oczekiwania na wciśnięcie dowolnego klawisza przez użytkownika;
- program ma informować na bieżąco użytkownika o wykonywanych operacjach w postaci informacji tekstowych wyświetlanych w oknie konsoli tekstowej;
- do obsługi wejścia/wyjścia i obsługi plików używać tylko strumieni, a do wszelkich łańcuchów znaków (tekstu) używać tylko typu string.

Szczegółowe wymagania dotyczące zadania:

- nazwa pliku wejściowego zawierającego model źródła informacji z rozszerzeniem ***.msort** ma być podawana jako parametr wejściowy przed uruchomieniem programu, plik wejściowy ma znajdować się w bieżącym katalogu roboczym;
- w wyniku działania programu w pliku wynikowym o nazwie takiej jak nazwa pliku wejściowego lub innej dowolnej, ale w każdym przypadku z rozszerzeniem ***.tree** ma być zapisane drzewo kodowania w postaci tabeli jednoznacznie opisującej drzewo. Tabela ma zawierać symbole podstawowe i zastępcze jako poszczególne węzły drzewa (tylko te, które mają potomków) wraz z informacją o ich połączeniach w formacie:

  symbol jako ojciec ***<spacja>*** *symbol jako potomek lewy* ***<spacja>*** *symbol jako potomek prawy*
  symbol jako ojciec ***<spacja>*** *symbol jako potomek lewy* ***<spacja>*** *symbol jako potomek prawy*
  ...
  symbol jako ojciec ***<spacja>*** *symbol jako potomek lewy* ***<spacja>*** *symbol jako potomek prawy*

  Symbole podstawowe to liczby naturalne w systemie dziesiętnym, a jako symbole zastępcze należy użyć kolejnych liczb całkowitych ujemnych w systemie dziesiętnym od **-1** w dół.

  Sposób sortowania modelu źródła przy kolejnych jego modyfikacjach zgodnie z algorytmem tworzenia drzewa kodowego ma być za każdym razem taki sam jak podczas wyznaczania pierwotnego modelu źródła czyli o kolejności decyduje liczba wystąpień w porządku **malejącym**, a gdy liczby wystąpień są takie same to decyduje wartość liczbowa symbolu (podstawowego lub zastępczego) w **porządku narastającym**.

Przy tworzeniu drzewa kodowania na podstawie modelu źródła ma być przyjęta zasada, że symbol ostatni w posortowanym modelu będzie **potomkiem lewym** w drzewie, a symbol przedostatni w posortowanym modelu będzie **potomkiem prawym** w drzewie.

- w wyniku działania programu w pliku wynikowym o nazwie takiej jak nazwa pliku wejściowego lub innej dowolnej, ale w każdym przypadku z rozszerzeniem ***.coding** ma być zapisana tabela kodu zawierająca symbole podstawowe i zastępcze wraz z przyporządkowanymi im łańcuchami znaków kodujących (typ string) składających się z „0" i „1" (utworzone w porządku **"0"** dla **lewego potomka**, **"1"** dla **prawego potomka**) w formacie:

  symbol podstawowy lub zastępczy ***<spacja>*** *łańcuch znaków kodujących z „0" i „1"*
  symbol podstawowy lub zastępczy ***<spacja>*** *łańcuch znaków kodujących z „0" i „1"*
  ...
  symbol podstawowy lub zastępczy ***<spacja>*** *łańcuch znaków kodujących z „0" i „1"*

  W pliku tym ma być także zapisany symbol zastępczy oznaczający korzeń drzewa wraz z przyporządkowanym mu pustym łańcuchem znaków.

- w wyniku działania programu w pliku wynikowym o nazwie takiej jak nazwa pliku wejściowego lub innej dowolnej, ale w każdym przypadku z rozszerzeniem ***.code** ma być zapisana tabela kodowa zawierająca tylko właściwe symbole podstawowe (bajty 8-bitowe) wraz z przyporządkowanymi im łańcuchami znaków kodujących (typ string) składających się z „0" i „1" w formacie:

  symbol (liczba naturalna w systemie dziesiętnym) ***<spacja>*** *łańcuch znaków kodujących z „0" i „1"*
  symbol (liczba naturalna w systemie dziesiętnym) ***<spacja>*** *łańcuch znaków kodujących z „0" i „1"*
  ...
  symbol (liczba naturalna w systemie dziesiętnym) ***<spacja>*** *łańcuch znaków kodujących z „0" i „1"*

  Można - ale nie jest to wymagane - dla zwiększenia efektywności kodowania posortować tabelę kodową przy zapisywaniu do pliku w takim porządku w jakim był na samym początku posortowany model źródła informacji.

- do sortowania zastosować funkcje **qsort()** lub **sort()** albo samodzielnie zaimplementowane sortowanie bąbelkowe lub inne;
- w programie poza funkcją **main()** mają być oddzielne funkcje z przekazywanymi do nich odpowiednimi argumentami co najmniej dla operacji wczytywania modelu źródła, wyznaczania drzewa kodowania oraz wyznaczania tabeli kodowej.

W wersji **podstawowej** jako wszelkie tablice należy odpowiednio zastosować typ danych tablica statyczna.

W wersji **zaawansowanej** jako wszelkie tablice należy zastosować listę dynamiczną zaimplementowaną samodzielnie lub skorzystać z gotowych kontenerów z biblioteki STL. W tej wersji można także zamiast odpowiednio utworzonej tablicy do przechowywania struktury drzewa kodowania zaimplementować strukturę danych pełnego drzewa binarnego korzystając ze wskaźników lub gotowych kontenerów z biblioteki STL.