# 20. Morseova abeceda (algoritmy převodu)

V Morseově abecedě jsou příslušné hodnoty krátký a dlouhý signál. V zápisu se nechá zapsat jako pomlčka a tečka. Dále na oddělení písmene /, na slovo // a na větu ///. Morseova abeceda vyplývá ze statistiky a pravděpodobnosti.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A | .– | M | – – | Y | –.– – | 6 | –…. |
| B | –… | N | –. | Z | – –.. | 7 | – –… |
| C | –.–. | O | – – – | Ä | .–.– | 8 | – – –.. |
| D | –.. | P | .– –. | Ö | – – –. | 9 | – – – –. |
| E | . | Q | – –.– | Ü | ..– – | . | .–.–.– |
| F | ..–. | R | .–. | Ch | – – – – | , | – –..– – |
| G | – –. | S | … | 0 | – – – – – | ? | ..– –.. |
| H | …. | T | – | 1 | .– – – – | ! | ..– –. |
| I | .. | U | ..– | 2 | ..– – – | : | – – –… |
| J | .– – – | V | …– | 3 | …– – | “ | .–..–. |
| K | –.– | W | .– – | 4 | ….– | ‘ | .– – – –. |
| L | .–.. | X | –..– | 5 | ….. | = | –…– |

## Implementace převodu textu na Morseovu abecedu

Metodě, která převede text na Morseovu abecedu, se předá String s textem, jenž se má převést. Poté se do nově vytvořeného řetězce přidává vždy kód daného znaku.

## Implementace převodu Morseovi abecedy na text

Metodě, která převádí Morseovu abecedu na obyčejný text, se předá hodnota jednoho Morseova písmene a ona vrátí jeho hodnotu v normální abecedě.

## Implementace pomocí enumu

Vytvoří se enum, kde budou všechny znaky (v normální a Morseově abecedě). Poté se při překladu vyhledávají příslušné znaky a zapisují se.

### Výhoda

* Primitivní
* Snadno realizovatelný

### Nevýhoda

* Pomalý (Vyhledávání)

#### public enum Abeceda {

#### A('a', ".-"),

…

#### NULA('0', "-----"),

…

#### 

#### private char c;

#### private String s;

#### Abeceda(char c, String s) {

#### this.c = c;

#### this.s = s;

#### }

#### public char getC() {

#### return c;

#### }

#### 

#### public String getS() {

#### return s;

#### }

## Implementace pomocí pole

Vytvoří se pole Stringů 🡪 pozice odpovídají znakům v ASCII tabulce (0 = ‘a‘…). Poté pomocí vzorce () 🡪 A je v ASCII tabulce na pozici 65. Když se od načteného znaku odečte 65, tak vyjde číslo mezi 0 a 25 🡪 index v daném poli.

#### public class TextToMorse{

#### public static final String morse[] = {".-", "-...", "-.-.", "-..",".", "..-.", "--.", "....", "..", ".---", "-.-", ".-..", "--", "-.", "---", ".--.", "--.-", ".-.", "...", "-", "..-", "...-", ".--", "-..-",

#### "-.--", "--..", ".----", "..---", "...--", "....-", ".....", "-....", "--...", "---..", "----.", "-----", "|"};

#### public static String textToTranslate = "Eis Was Here";

#### public static String transleText(String text, String[] array){

#### String finalSt = "";

#### for(int i = 0; i < text.length(); i++) {

#### char znak = text.charAt(i);

#### //Malá písmena

#### if((znak >= 'a') && (znak <= 'z')) {

#### finalSt += (morse[znak - 'a']);

#### }

#### if((znak >= 'A') && (znak <= 'Z')) {

#### //Velká písmena

#### finalSt += (morse[znak - 'A']);

#### }

#### finalSt += '/';

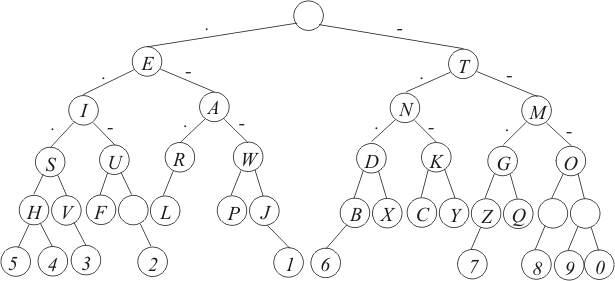
#### }

#### finalSt += '/';

#### return finalSt;

#### }

#### }

Při převodu morseovky na text se využívá binární strom. Nejvyšší prvek je mezera, která se dělí na ‘.‘ a ‘-‘. Při převodu se začíná na pozici 1 a postupně se čte znak po znaku, když je znak ‘.‘ vynásobí se aktuální pozice \* 2, když je znak ‘-‘ vynásobí se pozice \* 2 a přičte 1.

#### public class MorseToText{

#### public static final char chars[] = {'?', ' ', 'E', 'T', 'I', 'A', 'N', 'M', 'S', 'U', 'R', 'W', 'D', 'K', 'G', 'O','H', 'V', 'F', '?', 'L', '?', 'P', 'J', 'B', 'X', 'C', 'Y', 'Z', 'Q'};

#### public static String morseText =

#### "./../...//.--/.-/...//...././.-././";

#### public static String morseToText(String morse, char[] array){

#### String finalStr = "";

#### int position = 1;

#### for(int i = 0; i < morse.length(); i++) {

#### char character = morse.charAt(i);

#### switch(character) {

#### case '.':

#### position = position \* 2;

#### break;

#### case '-':

#### position = position \* 2 + 1;

#### break;

#### case '/':

#### finalStr += chars[position];

#### position = 1;

#### break;

#### }

#### }

#### finalStr += chars[position];

#### return finalStr;

#### }

#### }