

# Huffman encoder

## Obsah

1. Popis.....	1
2. Závislosti .....	2
3. Implementačné informácie .....	2
4. Piny .....	2
5. Konfigurácia .....	2
5.1 Konfiguračná štruktúra.....	2
5.2 Konfiguračný dialóg.....	3
6. Definícia triedy .....	3
7. Príklad použitia .....	4

## 1. Popis



Trieda *Huffman encoder* zakóduje pole celých nezáporných čísel do zhustenej binárnej podoby použitím Huffmanovho kódu. Analyzuje pole čísel, zistí početnosti prvkov poľa, vytvorí Huffmanov kód. Vytvorenými kódovými slovami zakóduje prvky poľa. Trieda *Huffman encoder* dokáže pracovať aj so zloženými znakmi (*N-gramy*), kde *N* po sebe idúcich prvkov poľa tvorí jeden znak. Ak počet prvkov poľa nie je násobok *N*, posledné zostávajúce prvky sa pridajú k znaku, ktorý má rovnaký prefix. Ak taký neexistuje, tak posledné prvky tvoria skrátený znak. Číslo *N* je možné nastaviť v konfigurácii.

Pole čísel je kódované nasledovne (počet bitov - položka) (znak a prvok nie je to isté):

32	<i>AllCount</i>	// počet prvkov vstupného poľa
32	<i>ItemCount</i>	// počet prvkov vo frekvenčnej tabuľke
8	<i>Ngrams</i>	// číslo <i>N</i>
8	<i>BitsToWord</i>	// počet bitov použitý na prvok poľa
8	<i>BitsToCount</i>	// počet bitov použitý na početnosť znaku

// tabuľka znakov, tento záznam sa opakuje *ItemCount* krát

```
{
BitsToWord  ItemOfArray      // prvok poľa zapísaný na minimálny počet bitov
}
```

// tabuľka početností, tento záznam sa opakuje toľko krát, koľko je znakov vo frekvenčnej tabuľke (teda (*ItemCount* div *Ngrams*)+1 krát)

```
{
BitsToCount  CountOfItem      // početnosť znaku vo frekvenčnej tabuľke
}
```

```
// bitový prúd kódových slov, tento záznam sa opakuje (ItemCount div Ngrams)+1 krát
{
BitsToCodeWord    CodeWord    // Ngramy kódované prislúchajúcimi kódovými slovami
}
```

## 2. Závislosti

Trieda *Huffman encoder* používa tieto externé definičné súbory a knižnice:

**Headers:** *filtergraph.h*, *datatypes.h*, *bitstream.h*

**Libs:** *filtergraph.lib*, *bitstream.lib*

## 3. Implementačné informácie

Informácie o triede:

**Názov:** *Huffman encoder*

**Verzia:** *1.0*

**Magic:** *190*

Informácie o definíciách:

**Header:** *huffmanencoder.h*

**Konfiguračný header:** *huffmanencodercfg.h*

**Lib:** *huffmanencoder.lib*

Informácie o knižnici obsahujúcej triedu:

**Názov knižnice:** *Huffman encoder Library*

**Verzia knižnice:** *1.0*

**Dll súbor knižnice:** *huffmanencoder.dll*

## 4. Piny

Piny sú popísané spôsobom: „*[index pinu na filtri] názov pinu: popis pinu*“.

**[0] Cfg in:** Konfiguračný vstup. Vstupom je serializovaná konfigurácia.

**[1] Cfg out:** Konfiguračný výstup. Výstupom je serializovaná konfigurácia.

**[2] In:** Dátový vstup. Vstupom je pole celých čísel zapísané v štruktúre *TIntArray*.

**[3] Out:** Dátový výstup. Výstupom je buffer binárnych dát zakódovaný hore uvedeným spôsobom.

## 5. Konfigurácia

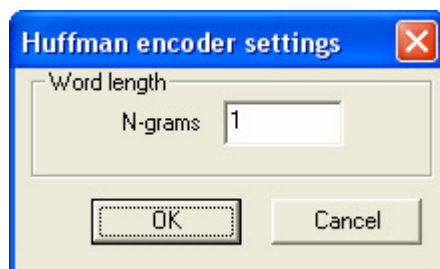
### 5.1 Konfiguračná štruktúra

Konfiguračná štruktúra je definovaná nasledovne:

```
typedef struct {
    int NGrams;           // Ngramy, počet prvkov poľa, ktorý tvorí jeden znak pre kódér
} THuffmanEncoderConfig;
```

V konfiguračnej štruktúre je možné nastaviť počet po sebe idúcich prvkov vstupného poľa, ktorý tvorí jeden znak pre kódér (*N-gram*). Štandardný Huffmanov kód používa *NGrams=1*.

## 5.2 Konfiguračný dialóg



Konfiguračný dialóg poskytuje voľbu počtu *NGrams* popísaného v konfiguračnej štruktúre.

## 6. Definícia triedy

Trieda *Huffman encoder* je definovaná nasledovne:

```
class THuffmanEncoderFilter : public TFilter {
private:
    HINSTANCE hInstance;           // identifikácia inštancie knižnice

    TIntArray *Input;              // vstupné pole celých čísel
    TBuffer    Output;             // výstupný buffer

    TPin* CfgInPin;                 // vstupný konfiguračný pin
    TPin* CfgOutPin;                // výstupný konfiguračný pin
    TPin* InPin;                    // vstupný pin
    TPin* OutPin;                   // výstupný pin

    bool StopFlag;                  // indikátor zastavenia

    CHuffmanEncoder *HuffmanEncoder; // inštancia huffmanovho kódéra

    int NGrams;                     // počet N

    int setConfigFromPin(TPin *pin); // nastaví konfiguráciu z pinu
    int putConfigOnPin(TPin *pin);   // sprístupní konfiguráciu na pine
    int freeConfigOnPin(TPin *pin);  // uvoľní konfiguráciu na pine
    int clearOutputPins();           // vyčistí výstupné piny
    int freeOutputData();            // uvoľní výstupné dáta
public:
    THuffmanEncoderFilter();         // konštruktor
    THuffmanEncoderFilter(HINSTANCE hInst); // konštruktor s parametrami
    ~THuffmanEncoderFilter();        // deštruktor

    int setConfigData(TBuffer config,int type); // nastaví konfiguráciu
    TBuffer getConfigData(int type);           // vráti konfiguráciu
    int initialize();                           // inicializácia
}
```

```

int run();
int reset();
int stop();
int finalize();
int showConfigDialog();
};

```

*// beh*  
*// reset stavu*  
*// ukončenie behu*  
*// finalizácia*  
*// vyvolá konfiguračný dialóg*

## 7. Príklad použitia

