**Lab 7**

**LZ77 coding algorithm**

**The purpose of the work** : to study the LZ77 algorithm used for lossless data compression, to master its principles and to implement the process of encoding and decoding messages. To learn how to work with character sequences and to evaluate the efficiency of the algorithm on different types of data.

**Theoretical part**

**The LZ77 algorithm** was developed by Abraham Lempel and Jacob Ziv in 1977 and is one of the first and most widely used lossless dictionary data compression algorithms. The basic idea behind LZ77 is that repeating character sequences are replaced by references to their previous occurrences, which allows for a significant reduction in data size.

LZ77 uses **a sliding window** that is divided into two parts:

* **Dictionary** - contains the already processed part of the data.
* **Buffer** - contains new characters that have not yet been processed.

**The main steps of the algorithm:**

1. The algorithm searches the dictionary for substrings that match the beginning of the buffer.
2. If a match is found, it is replaced with a triple of parameters:
   * *Offset* (where the match is in the dictionary);
   * *Length of the matching substring* (how many characters match);
   * *The next character* that comes after the match.
3. If no match is found, the algorithm encodes the character as <0, 0, character> .
4. The window is shifted by the length of the matching substring plus one character to continue processing the data.

**Example of work:**

Let's say we have a string: "ABABABA"

1. First match: symbol "A" not found in dictionary, encoded as <0, 0, A> .
2. The next character "B" is also not found in the dictionary, we encode it as <0, 0, B> .
3. The next match "AB" has already been found, we encode it as <2, 2, A> , where 2 is the offset (how many characters back we need to go), 2 is the length of the match, A is the character that follows the match.

The algorithm continues until all symbols have been processed.

**Advantages and disadvantages:**

* **Advantages** : The algorithm is effective for data with repeating sequences. Works well for text, graphics and binary data.
* **Disadvantages** : When the dictionary size is large, the speed of operation slows down, and encoding of single characters may be inefficient.

**Practical part**

Implement the LZ77 algorithm for data compression and decompression. The program should accept a string as input and return a compressed representation of the data as a sequence of codes <offset, length, symbol> .

**Task for all options:**

1. Implement the LZ77 algorithm for encoding. The search window and buffer length are set according to the variant.
2. Conduct experimental string compression.
3. Evaluate the effectiveness of the algorithm by comparing the amount of data before and after compression.
4. Before defending the lab, manually encode the given string (according to options) and compare with the encoding implemented programmatically.

**Options**

|  |  |  |  |
| --- | --- | --- | --- |
| **Option** | **Test line** | **Search window (length)** | **Buffer (length)** |
| 0 | ABABABAB | 5 | 3 |
| 1 | BA ABABA A | 4 | 3 |
| 2 | B A A A B D D | 6 | 3 |
| 3 | B BABABA A | 6 | 5 |
| 4 | BA ABAB DD | 6 | 4 |
| 5 | B A A A B D A | 4 | 3 |
| 6 | ABABABA A | 5 | 3 |
| 7 | BABA AB DA | 4 | 3 |
| 8 | BABA AB BA | 6 | 3 |
| 9 | BA ABABA D | 6 | 5 |