

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КІЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Комп'ютерний практикум №2
з курсу алгоритми кодування двійкових
даних

РЕАЛІЗАЦІЯ БІТОВОГО ПОТОКУ

Виконав студент
групи ФІ-42мн
Беш Радомир Андрійович

Київ — 2026

Зміст

1	Мета	3
2	Постановка задачі	3
3	Хід роботи	3
3.1	Особливості реалізації	3
3.2	Результати	5
4	Висновки	6

1 Мета

Опанувати методи роботи із потоками бітів та із масивами даних на рівні окремих бітів.

2 Постановка задачі

Реалізувати програмно інтерфейс для роботи із двійковими потоками даних. Реалізувати дві функції:

ReadBitSequence() - функція, яка вичитує із файла послідовність вказаної довжини;

WriteBitSequence() - функція, яка записує у файл послідовність бітів вказаної довжини.

Результати дослідження:

https://github.com/Radomir21/Encoding-Algorithms/tree/main/lab_2.

3 Хід роботи

3.1 Особливості реалізації

1. Створення класу для роботи з бітовим потоком

Було реалізовано окремий клас *BitStream*. Даний клас зберігає внутрішній байтовий буфер та позицію поточного біта, що дозволяє виконувати побітові операції читання і запису поверх звичайного файлу.

```
Encoding Algo > Encoding-Algorithms > lab_2 > bit_stream.py > ...
 1  class BitStream:
 2      def __init__(self, file, mode):
 3          self.file = file
 4          self.mode = mode
 5          self.buffer = 0
 6          self.bit_pos = 0
 7
 8      def close(self):
 9          if self.mode == 'w' and self.bit_pos != 0:
10              self.buffer <= (8 - self.bit_pos)
11              self.file.write(bytes([self.buffer]))
12          self.file.close()
13
```

2. Буферизація бітів під час запису

Запис бітів реалізовано шляхом поступового накопичення бітів у буфери. Кожен новий біт зсуває вміст буфера вліво та додається у молодший розряд. Після заповнення 8 біт формується байт і записується у файл.

```
15  def WriteBitSequence(bs, data, bit_length):
16      bit_index = 0
17
18  for _ in range(bit_length):
19      byte_index = bit_index // 8
20      bit_in_byte = 7 - (bit_index % 8)
21      bit = (data[byte_index] >> bit_in_byte) & 1
22
23      bs.buffer = (bs.buffer << 1) | bit
24      bs.bit_pos += 1
25
26  if bs.bit_pos == 8:
27      bs.file.write(bytes([bs.buffer]))
28      bs.buffer = 0
29      bs.bit_pos = 0
30
31      bit_index += 1
```

3. Зчитування з файлу

Якщо у буфері відсутні доступні біти, з файлу зчитується новий байт, з якого по черзі витягаються біти, починаючи зі старшого. Функція *ReadBitSequence()* дозволяє зчитувати послідовності довільної довжини, незалежно від їх вирівнювання відносно байтів у файлі.

```

34     def ReadBitSequence(bs, bit_length):
35         result = []
36         current_byte = 0
37         bits_collected = 0
38
39         for _ in range(bit_length):
40             if bs.bit_pos == 0:
41                 byte = bs.file.read(1)
42                 if not byte:
43                     raise EOFError("Кінець файлу")
44                 bs.buffer = byte[0]
45                 bs.bit_pos = 8
46
47                 bit = (bs.buffer >> (bs.bit_pos - 1)) & 1
48                 bs.bit_pos -= 1
49
50                 current_byte = (current_byte << 1) | bit
51                 bits_collected += 1
52
53                 if bits_collected == 8:
54                     result.append(current_byte)
55                     current_byte = 0
56                     bits_collected = 0
57
58                 if bits_collected > 0:
59                     current_byte <<= (8 - bits_collected)
60                     result.append(current_byte)
61
62         return bytes(result)

```

4. Вирівнювання бітового потоку при завершенні запису

Вирівнювання бітового потоку до межі байта виконується лише під час закриття файла. Якщо у буфері залишаються незаписані біти, вони доповнюються нульовими бітами та записуються у файл як завершальний байт. Даний механізм створен в класі *BitStream*.

3.2 Результати

<code>test.bin</code>	<code>t</code>	<code>...</code>	<code>t2.bin</code>	<code>...</code>	<code>t3.bin</code>	<code>...</code>	<code>t4.bin</code>	<code>...</code>	<code>t5.bin</code>	<code>...</code>
ing-Algorithms > lab_2 > <code>test.bin</code> coding Algo > Encoding-Algorithms > Encoding-Algorithms > lab_2 > <code>t4дин-Algorithms > lab_2 > <code>t5.bin</code></code>										
00 01 02 03 04 05 0 00 01 02 03 04 05 0 00 01 02 03 04 05 0 00 01 02 03 04 05 0 00 01 02 03 04 05 0	E1 77 00 +	00000000 A8 +	00000000 AA 55 +	00000000 F0 AA +	00000000 F0 0F AA +					

4 Висновки

У даній лабораторній роботі було реалізовано інтерфейс для побітового запису та зчитування даних. Реалізація дозволяє працювати з бітовими послідовностями довільної довжини без прив'язки до меж байта.