Отчёт по ЛР 1

Студент: **Либченко Михаил Вадимович**, **M4106**

Алгоритм: **Код Хаффмана двухпроходный, блоками по 2 символа**

Запуск

> Для запуска необходимо скачать `.exe`, соответствующий платформе запуска

```
./Archivator.exe [encoder|decoder] [inputPath] [outputPath]
```

Энкодер

Процесс кодирования

Сначала файл разбивается на пары байтов (2-байтовые блоки). Каждая пара интерпретируется как 16-битное число `ushort`. Для всех таких пар считается частота появления.

Для построения дерева используется приоритетная очередь (`PriorityQueue`). Каждая пара добавляется в очередь как лист дерева. На каждом шаге извлекаются 2 узла с наименьшими частотами и объединяются в нового родителя. Этот процесс повторяется до тех пор, пока не останется один корневой узел.

Проходимся по каждому символу *(паре байтов)* дерева и назначаем ей уникальную последовательность битов. Для левого ребра добавляется `0`, для правого - `1`.

Исходный файл снова разбивается на пары. Каждая пара заменяется соответствующим кодом Хаффмана. Последний одиночный байт *(если есть)* также кодируется как пара с `0x00`.

Полученная строка битов разбивается на байты. Если в последнем байте недостаточно бит, он дополняется нулями.

В выходной файл в указанном порядке записываются следующие данные:

- Количество уникальных пар
- Частотная таблица
- Оригинальная длина входного файла
- Длина сжатых данных
- Сами сжатые данные

Декодер

Процесс декодирования

Из архива считывается в указанном порядке следующие данные:

- Количество уникальных пар
- Частотная таблица
- Оригинальная длина входного файла
- Длина сжатых данных
- Сами сжатые данные

Аналогично кодеру, дерево восстанавливается из частотной таблицы.

Поток сжатых данных читается побитно. Для каждой последовательности битов происходит переход по дереву Хаффмана. Как только достигается лист *(восстановленная пара байтов)*, эта пара добавляется в выходной массив.

В процессе восстановления учитывается оригинальный размер входного файла. Если последний блок был неполным *(например, состоял только из одного байта + фиктивный `0x00`)*, то при декодировании второй байт просто не записывается.

Результаты сжатия

File	Entropy H(X)	Entropy H(X|X)	Entropy H(X|XX)	Avg bits/symbol	Initial size	Compressed size	Compressed (%)
bib	5.201	3.364	2.308	4.292	108,65 KB	58,29 KB	46.350
book1	4.527	3.585	2.814	4.070	750,75 KB	381,99 KB	49.119
book2	4.793	3.745	2.736	4.282	596,54 KB	319,3 KB	46.474

File	Entropy H(X)	Entropy H(X|X)	Entropy H(X|XX)	Avg bits/symbol	Initial size	Compressed size	Compressed (%)
geo	5.646	4.264	3.458	4.608	100 KB	57,6 KB	42.396
news	5.190	4.092	2.923	4.650	368,27 KB	214,05 KB	41.878
obj1	5.948	3.464	1.400	4.585	21 KB	12,04 KB	42.685
obj2	6.260	3.870	2.265	4.465	241,03 KB	134,53 KB	44.184
paper1	4.983	3.646	2.332	4.319	51,92 KB	28,02 KB	46.019
paper2	4.601	3.522	2.514	4.064	80,27 KB	40,78 KB	49.198
paper3	4.665	3.555	2.560	4.114	45,44 KB	23,37 KB	48.569
paper4	4.700	3.477	2.205	4.065	12,97 KB	6,59 KB	49.187
paper5	4.936	3.526	2.041	4.217	11,67 KB	6,15 KB	47.281
paper6	5.010	3.611	2.251	4.307	37,21 KB	20,04 KB	46.157
pic	1.210	0.824	0.705	1.193	501,19 KB	74,73 KB	85.090
progc	5.199	3.603	2.134	4.400	38,68 KB	21,27 KB	45.003
progl	4.770	3.212	2.044	4.001	69,97 KB	34,99 KB	49.992
progp	4.869	3.188	1.755	4.028	48,22 KB	24,28 KB	49.645
trans	5.533	3.355	1.930	4.452	91,5 KB	50,92 KB	44.346

> Суммарное значение сжатого размера всех файлов: **1_545_175 байт**