# Лабораторная работа №2: создание архиватора

## Цели

1. Отработать на практике работу с файлами.
2. Научиться реализовывать консольную утилиту средствами .NET.

## Задание

Написать программу-архиватор, выполняющую сжатие/извлечение указанного ей файла.

Формат сжимаемых данных и алгоритм сжатия остаётся на усмотрение студента: допускается применение любых алгоритмов и форматов (в том числе можно придумать свой). Нельзя использовать готовые библиотеки для сжатия данных.

Программа должна быть реализована в виде консольного приложения, получающего три параметра из командной строки:

* Признак действия с файлом (сжатие/извлечение).
* Название исходного файла;
* Название целевого файла;

Программа должна прочитать исходный файл и в зависимости от признака действия сжать файл или распаковать его, сохранив результат на диске в целевом файле.

Пример строки запуска (запуск из командной консоли Windows, программа должна распаковать файл archive.bin, ранее созданный этой же утилитой, в textfile.txt):

Packer.exe –unpack archive.bin textfile.txt

Выполнение перечисленных выше требований оценивается как «удовлетворительно». Для получения более высоких оценок необходимо дополнительно:

* Для получения оценки «Хорошо» выполнить все следующие требования:
  + отображать время работы программы в секундах и коэффициент сжатия при выполнении сжатия (отношение размера упакованного файла к исходному);
  + добавить параметр командной строки “test” в программу, при указании которого она будет формировать случайный тестовый файл, на котором можно проверить сжатие. Размер файла (можно примерный размер, а не точный) должен задаваться вторым байтом. Пример вызова:

Packer.exe –testfile.txt 100000

* Для получения оценки «Отлично» необходимо к требованиям оценки «хорошо» выполнить следующее требование:
  + программа должна сжимать файл размером 100 Мб не дольше 50 секунд на студенческом компьютере в лаборатории.

## Демонстрация работы

1. Рассказать коротко используемые алгоритмы сжатия и извлечения.
2. Подготовить два файла: хорошо сжимаемый разработанным архиватором, и плохо сжимаемый. Пояснить, почему один из файлов сжимается хорошо, а другой плохо.
3. Сжать оба файла. Размер одного из архивов должен оказаться меньше размера исходного файла.
4. Извлечь оба файла из архива. Должны получиться исходные файлы.
5. Для проверки на оценку отлично необходимо подготовить тестовый файл размером не менее 100 Мб и сжать его.

## Вспомогательные материалы

### Алгоритмы сжатия

Алгоритмы сжатия делятся на две большие группы:

1. Сжатие без потерь;
2. Сжатие с потерями.

Сжатие без потерь означает, что после извлечения получаются точно те же данные, которые были сжаты. Сжатие с потерями подразумевает небольшие отклонения от исходных сжатых данных (допустимо, например, при сжатии звуков или изображений).

Примеры алгоритмов сжатия без потерь:

* RLE – замена одинаковых подряд идущих символов на указание числа символов и повторяемого символа (пример: XXYYZZZ -> 2X2Y3Z);
* Кодирование Хаффмана – построение дерева вероятностей распределения символов с последующей записью для каждого символа пути в дереве;
* Алгоритмы Лемпеля-Зива (LZ) – создание словаря и замена ранее встречавшихся в тексте слов на словарные .

Пример алгоритма сжатия с потерями:

* JPEG – дискретное косинус-преобразование с отбросом малозначащих чисел.

### Полезные методы .NET

* Чтение массива байт из файла: System.IO.File.ReadAllBytes;
* Запись массива байт в файл: System.IO.File.WriteAllBytes;
* Побайтовая работа с файлом: System.IO. FileStream;
* Атрибуты файла (в том числе размер): System.IO.FileInfo;
* Текущее время: System.DateTime.Now;
* Разница между двумя временами end и start в миллисекундах: end.Subtract ( start ).Milliseconds
* Параметры командной строки передаются в качестве аргумента точки входа программы: public static void Main(string[] args)

Пример побайтового чтения и записи в файл:

using ( var inputFile = new FileStream ( @"d:\infile.txt", FileMode.Open ) ) {

using ( var outputFile = new FileStream ( @"d:\outfile.txt", FileMode.Create ) ) {

while ( true ) {

int inByte = inputFile.ReadByte ( );

if ( inByte < 0 )

break;

outputFile.WriteByte ( (byte)inByte );

}

}

}