Состав приложения:

* Класс Program – точка входа
* Класс Worker – родительский класс для классов архивации и разархивации, содержит общие поля и описания методов
* Класс Compressor – содержит методы для архивации
* Класс Decompressor – содержит методы для разархивации
* Класс Extension – дополнительный статический класс для расширяющих методов.

Классы Compressor и Decompressor содержат следующие переопределенные методы:

* Start () – последовательное получение данных из исходного файла, разбиение на части и помещение их в коллекцию
* ProcessChunk (int chunkNumber) – получение блока необработанных данных из одной коллекции, обработка (архивация/разархивация) и помещение в другую коллекцию
* FlushToDisk() – последовательная запись обработанных блоков в файл

Краткое описание алгоритма архивации.

* Из исходного файла данные читаются в цикле блоками, размер которых определяется исходя из объема свободной оперативной памяти. Если объем свободной памяти составляет меньше 20% от первоначального объема свободной памяти – чтение приостанавливается и запускается принудительный сбор мусора 2-го поколения. Таким образом поддерживается постоянный запас свободной памяти.
* Считанные данные помещаются в словарь входящих данных, ключом которого является номер блока данных. Затем в отдельном потоке запускается сжатие блока. Количество параллельно исполняемых потоков сжатия контролируется с помощью семафора (устанавливается равным количеству логических процессоров).
* Архивация блока данных производится помещением его в поток типа GZipStream. После этого сжатый блок помещается в словарь исходящих данных и удаляется из словаря входящих данных. Многопоточный доступ к коллекциям блокируется с помощью оператора lock. При завершении сжатия первого блока данных запускается запись сжатых блоков в файл архива в отдельном потоке.
* Запись сжатых блоков в файл осуществляется в цикле в том же порядке, в котором они были считаны, благодаря сохраняющемуся порядковому номеру в виде ключа словаря. Если очередной блок не готов – поток приостанавливается. При записи сжатого блока в файл к его заголовку, сформированному GZipStream, добавляется размер блока для корректного разбиения при распаковке. Размер блока записывается, начиная с 4-го байта, в область, предназначенную для записи времени изменения (MTIME в соответствии с RFC 1952), не используемую GZipStream. Такой подход позволяет распаковывать архив в дальнейшем также сторонними средствами.

Разархивация происходит аналогично. При считывании блока сжатых данных из файла сначала считывается заголовок Gzip (первые 8 байт), из которого извлекается предварительно записанный размер всего блока. Затем исходя из полученного размера считывается оставшаяся часть блока. Таким образом, в цикле считываются последовательно все блоки сжатых данных. Дальнейший процесс аналогичен алгоритму архивации. Поскольку размер сжатого блока непредсказуем, при архивации устанавливается ограничение на максимальный размер исходных данных (256 МБ).

Для запуска программы используются параметры в соответствии с заданием. Также добавлен четвертый необязательный параметр [verbose], позволяющий выводить подробную информацию о ходе выполнения на экран:

* GZipTest.exe compress [имя исходного файла] [имя архива] [verbose]

В любом случае, подробная информация сохраняется в файл [имя исходного файла].log. Также записывается диагностическая информация при возникновении исключения.

Нажатие комбинации Ctrl-C обрабатывается с помощью делегата в базовом классе, закрывающего открытые потоки чтения/записи файлов и выставляющего флаг StopRequested, при наличии которого прекращаются итерации в запущенных циклах и предотвращается запуск новых потоков архивации/разархивации.