Программирование на языке Си

Практикум: Компрессор по методу Хаффмана (Часть 2)

Штанюк А.А.

6 февраля 2014 г.

Кодирование

В предыдущей части были рассмотрены:

1. Теоретические основы компрессии.
2. Алгоритм сжатия Хаффмана.
3. Структурная схема компрессора.
4. Блок анализа входного файла и генератор префиксного кода.

Следующим важнейшим элементом программы является блок кодирования. Суть кодирования заключается в том, что каждому символу (байту) входного файла сопоставляется строка с префиксным кодом из таблицы встречаемости и на этой основе формируется файл, целиком состоящий из символов '0' и '1', представляющий собой закодированное содержимое исходного (сжимаемого) файла.

Кодирование

Файл с закодированным содержимым (.101) не является обязательным при работе компрессора. В нашей работе он создаётся для промежуточного хранения закодированного содержимого исходного файла и может служить средством, упрощающим отладку всей программы. Этот файл может потребовать значительный объём памяти. После завершения работы компрессора он может быть удалён.

Нам необходимо прочитать исходный файл посимвольно в цикле до тех пор, пока не будет достигнут конец файла (EOF. Для того, чтобы EOF был распознан корректно, необходимо, чтобы функция чтения (**fgetc**) возвращала значение типа **int**.

Кодирование

Каждый прочитанный символ просматривается в массиве структур **SYM** в поле **ch**. После того, как совпадение произошло, необходимо вывести в .101-файл содержимое поля **code**.

```
int ch;        // код символа из файла
FILE *fp_in;   // исходный файл
FILE *fp_101; // закодированный файл
...
while((ch=fgetc(fp_in))!=-1)
{
    for(i=0;i<count;i++)
       if(syms[i].ch==(unsigned char)ch) {
           fputs(syms[i].code,fp_101);   // выводим строку с кодом
           break;                        // прерываем поиск
    }
}
```

Упаковка данных

Суть упаковки заключается в переводе значений строки, состоящей из '0' и '1', в значения двоичных разрядов байта. Данные для упаковки берутся из .101-файла, созданного на этапе кодирования.

```
union CODE {
   unsigned char ch;
   struct {
      unsigned short b1:1;
      unsigned short b2:1;
      unsigned short b3:1;
      unsigned short b4:1;
      unsigned short b5:1;
      unsigned short b6:1;
      unsigned short b7:1;
      unsigned short b8:1;
  } byte;
};
```

Если в каждый разряд поля битов **byte**, находящегося внутри объединения **CODE** записать 0 или 1, в зависимости от значения символа '0' или '1' из .101-файла, то в поле **ch** окажется байт с нужной информацией. Этот байт выводится в сжатый (результирующий) файл.

Функция упаковки

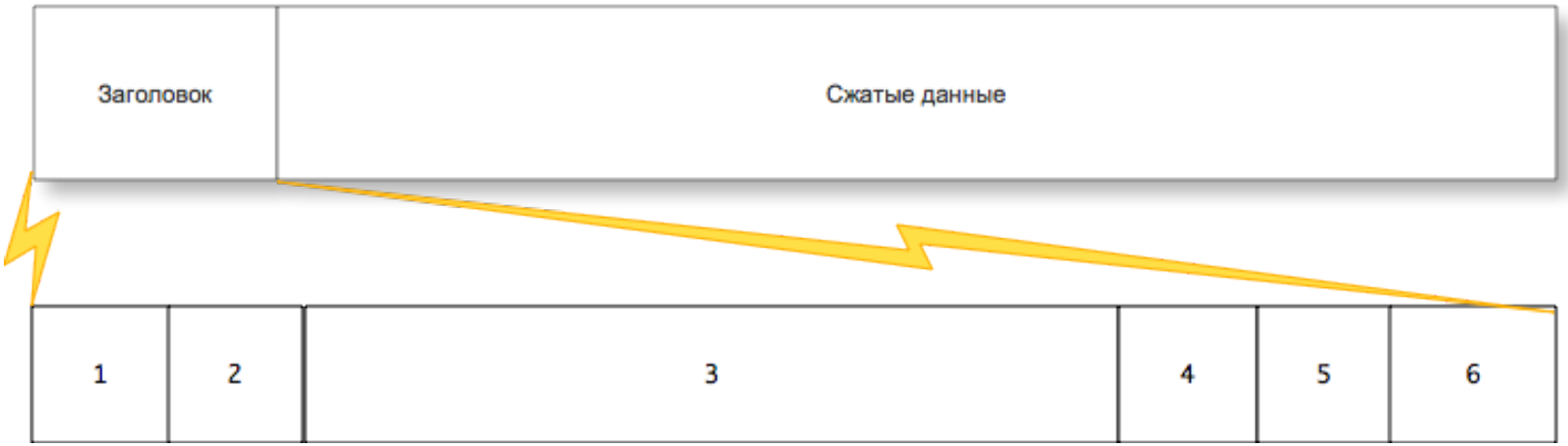
Пример функции, производящей упаковку:

```
unsigned char pack(unsigned char buf[])
{
  union CODE code;
  code.byte.b1=buf[0]-'0';
  code.byte.b2=buf[1]-'0';
  code.byte.b3=buf[2]-'0';
  code.byte.b4=buf[3]-'0';
  code.byte.b5=buf[4]-'0';
  code.byte.b6=buf[5]-'0';
  code.byte.b7=buf[6]-'0';
  code.byte.b8=buf[7]-'0';
  return code.ch;
}
```

На вход подаётся массив из 8 символов, среди которых могут быть '0' и '1'. Результатом работы является один сжатый байт.

Формирование заголовка

Для возможности обратного преобразования содержимого сжатого файла (декомпрессия) необходимо сформировать в сжатом файле заголовок.



1. сигнатура (подпись формата)
2. количество уникальных символов (число строк в таблице встречаемости)
3. сама таблица (пары: код символа - частота)
4. длина "хвоста" (остаток деления размера .101-файла на 8)
5. размер исходного файла (для проверки после декомпрессии)
6. расширение исходного файла (для восстановления)

Формирование заголовка

- Сигнатура. Набор из нескольких символов, позволяющий установить принадлежность файла определённому формату. Декомпрессор по этим символом должен "узнать" свой файл среди чужих.
- Число уникальных символов. Содержит количество строк в таблице встречаемости. Декомпрессор организует цикл для чтения всей таблицы.
- Таблица встречаемости. Здесь хранятся записи о символах и частотах. Хранить их нужно в бинарном виде без промежутков.
- Длина хвоста. Размер файла .101 не всегда кратен 8, поэтому в конце может появится "хвост" из нескольких битов. Мы дополняем их до полного байта и сохраняем актуальную длину.
- Размер исходного файла. Нужен для контроля после распаковки.
- Исходное расширение файла. Если компрессор меняет расширение исходного файла, то его нужно восстановить.

Вопрос о быстродействии

Время работы компрессора зависит от нескольких ключевых факторов:

- время анализа входного файла при построении таблицы встречаемости;
- время сканирования входного файла при кодировании;
- поиск в массиве структур при кодировании;
- время формирования .101-файла
- время обработки .101-файла при упаковке;
- время формирования сжатого файла.

Очевидно, что замедляют работу программы многократные проходы по содержимому исходного файла. Как вариант решения проблемы можно предложить загрузку всего содержимого файла в оперативную память (динамический массив) в самом начале, а затем производить сканирование динамического массива.

Поиск по массиву структур (стр. 8-12 в предыдущем листинге) не занимает много времени, поскольку массив отсортирован по частоте и записи о наиболее "популярных" символах находятся близко к началу массива.