Восьмеричный дамп целого числа

Задание. Для заданного короткого целого числа построить его восьмеричный дамп.

Проблема. Представить описание алгоритма, позволяющего выполнить это задание.

Обсуждение хода решения:

1) ввести число;
2) сформировать символьную последовательность на основе символьных литералов от ***'0'*** до ***'7'***, сохраняя в ней ведущие нули;
3) сформировать символьную последовательность на основе символьных литералов от ***'0'*** до ***'7'***, “подавляя” в ней ведущие нули.

Циклическая операция маскирования позволяет последовательно выявить значения для всех триад (восьмеричных цифр) из двоичных разрядов числа, начиная с триады, следующей за знаковым двоичным разрядом (неполной триадой, состоящей всего из одного своего младшего двоичного разряда). Знаковый двоичный разряд (старшую восьмеричную цифру), как видим, следует рассматривать независимо от остальных. Начальное значение маски можно задать с помощью восьмеричного литерала ***070000*** или шестнадцатеричного литерала ***0x7000***. Маску, как видим, всякий раз следует сдвигать вправо на три двоичных разряда. Для представления коротких целых чисел будем опираться на 16-разрядную архитектуру.

Обратимся к восьмеричному дампу короткого целого числа 13_{10}, строя его на основе анализа значений маскируемых младших восьмеричных разрядов числа с последующим арифметическим сдвигом вправо на три двоичных разряда маски:

Десятичный код	Восьмеричный код	Двоичный код	Восьмеричный дамп
13	000015	0000000000001101	0
13	000015	0000000000001101	0
	Маска	Маска	
	070000	0111000000000000	
13	000015	0000000000001101	0
	Маска	Маска	
	007000	0000111000000000	
13	000015	0000000000001101	0
	Маска	Маска	
	000700	0000000111000000	
13	000015	0000000000001101	1
	Маска	Маска	
	000070	0000000000111000	
13	000015	0000000000001101	5
	Маска	Маска	
	000007	0000000000000111	

Обратимся к восьмеричному дампу короткого целого числа -13_{10}, строя его на основе анализа значений маскируемых младших восьмеричных разрядов числа с последующим арифметическим сдвигом вправо на три двоичных разряда маски:

Десятичный код	Восьмеричный код	Двоичный код	Восьмеричный дамп
–13	177763	1111111111110011	1
–13	177763	1111111111110011	7
	Маска	Маска	
	070000	0111000000000000	
–13	177763	1111111111110011	7
	Маска	Маска	
	007000	0000111000000000	
–13	177763	1111111111110011	7
	Маска	Маска	
	000700	0000000111000000	
–13	177763	1111111111110011	6
	Маска	Маска	
	000070	0000000000111000	
–13	177763	1111111111110011	3
	Маска	Маска	
	000007	0000000000000111	

Описания алгоритмов построения восьмеричных дампов.

При разработке алгоритмов построения восьмеричных дампов коротких целых чисел будем опираться на арифметический сдвиг вправо маски. Чтобы символьная последовательность при этом формировалась в прямом порядке, следует использовать циклическую операцию маскирования исходного числа, которая позволит последовательно выявить значения всех его восьмеричных разрядов (триад), начиная с триады, следующей за знаковым двоичным разрядом (неполной триадой, состоящей всего из одного своего младшего двоичного разряда). Знаковый двоичный разряд (старшую восьмеричную цифру) следует рассматривать независимо от остальных. Начальное значение маски следует задать с помощью восьмеричного литерала ***070000*** или шестнадцатеричного литерала ***0x7000***. Маску всякий раз следует сдвигать вправо на три двоичных разряда. В основу механизма преобразования триады в восьмеричный символ (цифры от ***0*** до ***7***) кладётся функция ***char()***, аргументом которой является увеличенный на 48 двоичный код триады.

Представим структурированное описание алгоритма на естественном языке:

1. Начало
2. Повторить
2.1. Вывести значение “N? ”
2.2. Ввести значение N
пока не будет (N >= –32768) и (N <= 32767)
3. Положить M = 070000
4. Если N < 0
то
4.1. Вывести значение ‘1’
иначе
4.2. Вывести значение ‘0’
5. От k = 0 до k = 4 повторить
5.1. Вывести значение char(((N and M) asr 15 – 3 * k – 3) + 48)
5.2. Положить M = M asr 3
6. Конец

Теперь зададимся целью реализовать механизм подавления ведущих нулей для неотрицательных чисел. В основу этого механизма кладётся флаг – переменная, ненулевое значение которой “просигналит” о появлении первой “ведущей” цифры, отличной от нуля, в символьной последовательности. Нулевое значение флага препятствует появлению в символьной последовательности литералов ***'0'***, трактуя их как ведущие нули. Очевидно, что сам нуль станет теперь “особым” числом, которое следует рассматривать независимо от остальных неотрицательных чисел.

Представим структурированное описание алгоритма на естественном языке (версия 1):

1. Начало
2. Повторить
2.1. Вывести значение “N? ”
2.2. Ввести значение N
пока не будет (N >= –32768) и (N <= 32767)
3. Положить M = 070000
4. Положить flag = false
5. Если N не равно 0
то
5.1. Если N < 0
то
5.1.1. Положить flag = true
5.1.2. Вывести значение ‘1’
5.2. От k = 0 до k = 4 повторить
5.2.1. Если N and M не равно 0
то
5.2.1.1. Положить flag = true
5.2.1.2. Вывести значение char(((N and M) asr 15 – 3 * k – 3) + 48)
иначе
5.2.1.3. Если flag = true
то
5.2.1.3.1. Вывести значение ‘0’
5.2.2. Положить M = M asr 3
иначе
5.3. Вывести значение ‘0’
6. Конец

Представим структурированное описание алгоритма на естественном языке (версия 2):

1. Начало
2. Повторить
2.1. Вывести значение “N? ”
2.2. Ввести значение N
пока не будет (N >= –32768) и (N <= 32767)
3. Положить M = 070000
4. Положить flag = false
5. Если N не равно 0
то
5.1. Если N < 0
то
5.1.1. Положить flag = true
5.1.2. Вывести значение ‘1’
5.2. От k = 0 до k = 4 повторить
5.2.1. Положить D = (N and M) asr 15 – 3 * k – 3
5.2.2. Если D не равно 0
то
5.2.2.1. Положить flag = true
5.2.2.2. Вывести значение char(D + 48)
иначе
5.2.2.3. Если flag = true
то
5.2.2.3.1. Вывести значение ‘0’
5.2.3. Положить M = M asr 3
иначе
5.3. Вывести значение ‘0’
6. Конец