Практическая работа №2.1

«Сортировка числового файла с помощью битового массива»

**Цель работы**: освоить приёмы работы с битовым представлением беззнаковых целых чисел, реализовать эффективный алгоритм внешней сортировки на основе битового массива.

1. Битовое представление целых чисел и множеств чисел. Битовые операции в С++.

В языке программирования С++ предусмотрено несколько целочисленных типов данных: bool (он же логический), char (он же символьный), short int (чаще просто short), long int (он же int или long) и long long int (или просто long long). Числа этих типов занимают в памяти компьютера по 1, 2, 4 и 8 байт соответственно (см. табл. 1).

Значения всех этих типов бывают знаковые (signed) и беззнаковые (unsigned). В первом случае диапазон допустимых значений каждого из названных типов включает в себя как положительные, так и отрицательные числа. Во втором случае – только неотрицательные.

Таблица 1. Диапазоны значений числовых типов данных в языке С++.

|  |  |  |
| --- | --- | --- |
| **Тип данных** | **Диапазон значений** | **Размер (байт)** |
| bool | true, false (1, 0) | 1 |
| signed char | -128…127 | 1 |
| unsigned char | 0…255 | 1 |
| signed short int | -32768…32767 | 2 |
| unsigned short int | 0…65535 | 2 |
| signed long int | -2 147 483 648…2 147 483 647 | 4 |
| unsigned long int | 0…4 294 967 295 | 4 |
| signed long long int | -9 223 372 036 854 775 808…  9 223 372 036 854 775 807 | 8 |
| unsigned long long int | 0…18 446 744 073 709 551 615 | 8 |
| float | 3.4e-38…3.4e+38 | 3 |
| double | 1.7e-308…1.7e+308 | 8 |
| long double | 3.4e-4932…3.4e+4932 | 10 |

Разница диапазонов является следствием способа хранения целых чисел в памяти ЭВМ современных архитектур. Конечно, целое число в памяти хранится как битовая последовательность той длины, которая предусмотрена тем или иным типом.

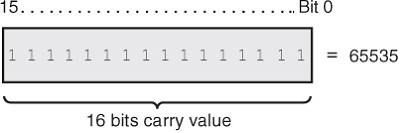


Рис. 1. Беззнаковое двухбайтовое целое число в памяти ЭВМ.

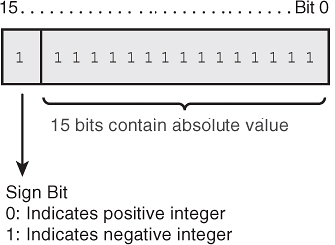


Рис. 2. Двухбайтовое целое число со знаком в памяти ЭВМ.

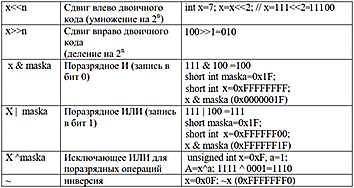
В беззнаковом типе все двоичные разряды (биты) отведены под абсолютное значение (модуль) числа (рис. 1).

В числе со знаком под модуль отведены все двоичные разряды, кроме старшего (рис. 2). Одно из значений старшего бита интерпретируется как знак «плюс», противоположное – как «минус». Т.к. разрядов под модуль числа на 1 меньше, то и наибольшее допустимое значение в типе со знаком вдвое меньше такового в беззнаковом типе.

**Примечание**: векторный способ организации числовых последовательностей (т.е. массивы чисел) в памяти компьютера формирует непрерывную последовательность бит от начального до конечного элемента этого массива.

При работе с битовыми представлениями чисел можно использовать ***битовые операции***, определённые в языке С++ (см. табл. 2).

Таблица 2. Битовые операции в С++.



Пример – как установить 5-й бит произвольного целого числа в 0 и что получится в результате:

*unsigned char x=255; //8-разрядное двоичное число 11111111*

*unsigned char maska = 1; //1=00000001 – 8-разрядная маска*

*x = x & (~ (maska<<4)); //результат x=239*

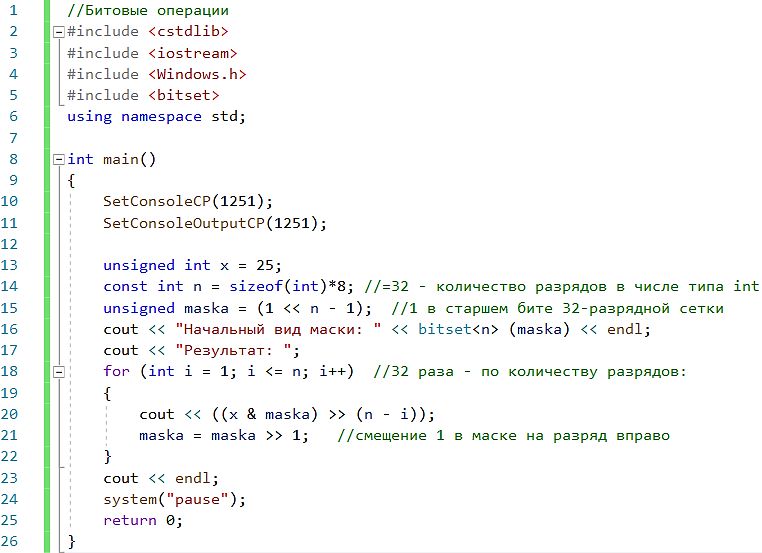
Задание 1:

**1.а.** Реализуйте вышеприведённый пример, проверьте правильность результата в том числе и на других значениях х.

**1.б.** Реализуйте по аналогии с предыдущим примером установку 7-го бита числа в единицу.

**1.в.** Реализуйте код листинга 1, объясните выводимый программой результат.

Листинг 1.



**Примечание**: как видно из примера (строка 16 листинга 1), битовый массив можно организовать и другими способами: с помощью класса bitset или класса vector из элементов типа bool[[1]](#footnote-1).

2. Сортировка последовательности чисел с помощью битового массива.

Пусть даны не более 8 чисел со значениями от 0 до 7, например, {1, 0, 5, 7, 2, 4}.

Подобный набор чисел удобно отразить в виде 8-разрядной битовой последовательности **11101101**. В ней единичные биты показывают ***наличие*** в исходном наборе числа, равного номеру этого бита в последовательности (нумерация с 0 слева). Т.о. индексы единичных битов в битовом массиве – это и есть числа исходной последовательности.

Последовательное считывание бит этой последовательности и вывод индексов единичных битов позволит естественным образом получить исходный набор чисел ***в отсортированном виде*** – {0, 1, 2, 4, 5, 7}.

В качестве подобного битового массива удобно использовать беззнаковое однобайтовое число (его двоичное представление в памяти), например, типа unsigned char. Приёмы работы с отдельными битами числа были рассмотрены в предыдущем задании.

Задание 2:

**2.а.** Реализуйте вышеописанный пример с вводом произвольного набора до 8-ми чисел (со значениями от 0 до 7) и его сортировкой битовым массивом в виде числа типа unsigned char. Проверьте работу программы.

Если количество чисел в исходной последовательности больше 8 и/или значения превосходят 7, можно подобрать тип беззнакового числа для битового массива с подходящим размером разрядной сетки – до 64 в типе unsigned long long (см. табл. 1).

**2.б.** Адаптируйте вышеприведённый пример для набора из 64-х чисел (со значениями от 0 до 63) с битовым массивом в виде числа типа unsigned long long.

Если количество чисел и/или их значения превосходят возможности разрядной сетки одного беззнакового целого числа, то можно организовать линейный массив (вектор) таких чисел, который в памяти ЭВМ будет представлен ***одной непрерывной битовой последовательностью***.

**2.в.** Исправьте программу задания 2.б, чтобы для сортировки набора из 64-х чисел использовалось не одно число типа unsigned long long, а линейный массив чисел типа unsigned char.

3. Быстрая сортировка числового файла с помощью битового массива.

На практике может возникнуть задача внешней сортировки, т.е. упорядочения значений, расположенных во внешней памяти компьютера, размер которых превышает допустимый объём ОЗУ (например, 1 МБ стека, выделяемый по умолчанию программе операционной системой).

Возможный способ – это алгоритм внешней сортировки слиянием, рассмотренный в одной из предыдущих практических работ. Считывание исходного файла при этом происходит один раз, но в процессе сортировки создаются и многократно считываются вспомогательные файлы, что существенно снижает быстродействие.

Второй возможный приём – считывание входного файла порциями, размер каждой из которых не превышает лимит ОЗУ. Результат записывается в выходной файл за один раз, при этом не используются вспомогательные файлы. Программа будет работать быстрее, но всё-таки есть алгоритм, существенно превосходящий перечисленные.

Реализовать высокоэффективную сортировку большого объёма числовых данных в файле можно на идее битового массива. Достаточно один раз считать содержимое файла, заполнив при этом в памяти ЭВМ битовый массив и на его основе быстро сформировать содержимое выходного файла в уже отсортированном виде.

При использовании битового массива для представления сортируемых чисел, программу можно представить как последовательность из трех подзадач:

а) Создание битового массива с нулевыми исходными значениями.

б) Считывание целых чисел из файла и установка в 1 соответствующих бит массива.

в) Формирование упорядоченного выходного файла путём последовательной проверки бит массива и вывода в файл номеров (индексов) тех бит, которые установлены в 1.

Задание 3.

Постановка задачи:

***Входные данные***: файл, содержащий не более n=107 неотрицательных целых чисел[[2]](#footnote-2), среди них нет повторяющихся.

***Результат***: упорядоченная по возрастанию последовательность исходных чисел в выходном файле.

***Время работы программы***: ~10 с (до 1 мин. для систем малой вычислительной мощности).

***Максимально допустимый объём ОЗУ*** для хранения данных: 1 МБ.

Очевидно, что размер входных данных гарантированно превысит 1МБ (это, к примеру, максимально допустимый объём стека вызовов, используемого для статических массивов).

Требование по времени накладывает ограничение на количество чтений исходного файла.

**3.а.** Реализуйте задачу сортировки числового файла с заданными условиями. Добавьте в код возможность определения времени работы программы.

**Примечание**: содержимое входного файла должно быть сформировано неповторяющимися значениями заранее, это время не должно учитываться при замере времени сортировки.

В отчёт внесите результаты тестирования для наибольшего количества входных чисел, соответствующего битовому массиву длиной 1МБ.

**3.б.** Определите программно объём оперативной памяти, занимаемый битовым массивом.

Содержание отчёта:

1. Титульный лист.
2. Цель работы.
3. Ход работы (по каждому заданию):
   1. Формулировка задачи.
   2. Математическая модель решения (описание алгоритма).
   3. Код программы с комментариями.
   4. Результаты тестирования.
4. Вывод (решены ли задачи, достигнута ли цель).

Для сдачи практической работы потребуется:

- отчёт – оформляется в виде электронного документа в форматах Word или PDF, прикрепляется к соответствующему заданию в СДО;

- программные проекты, реализованные по заданиям;

- доклад по результатам выполнения практической работы (по отчёту).

Вопросы для самоподготовки:

1. Что такое информация? Что такое 1 бит информации?
2. Как выделение одного бита под знак числа изменяет диапазон допустимых значений в числовых типах?
3. Что такое инверсия?
4. Зачем в программе из листинга 1 при циклическом выводе в консоль значений очередного бита (строка 20), его нужно смещать в младший разряд?
5. В примере задания 2.а могут ли входные значения быть за границами диапазона от 0 до 7?
6. Какое наибольшее количество входных чисел можно отсортировать с помощью битового массива длиной 1МБ?
7. Чем можно объяснить возможные отличия во времени выполнения сортировки одного и того же входного файла при нескольких запусках?
8. Предложите варианты доработки алгоритма сортировки битовым массивом на случай повторяющихся значений во входной последовательности.

Литература:

1. Страуструп Б. Программирование. Принципы и практика с использованием C++. 2-е изд., 2016.
2. Документация по языку С++ [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/cpp/cpp/> (дата обращения 01.09.2021).
3. Курс: Структуры и алгоритмы обработки данных. Часть 2 [Электронный ресурс]. URL: <https://online-edu.mirea.ru/course/view.php?id=4020> (дата обращения 01.09.2021).

1. Под значение типа bool выделяется 1 байт (8 бит) памяти, но в классе vector происходит оптимизация, в результате которой одно логическое значение занимает 1 бит. [↑](#footnote-ref-1)
2. Для упрощения кода работы с файлами в текстовом режиме можно ограничиться только семизначными числами в диапазоне [1000000…9999999]; для работы с файлами в бинарном режиме это не актуально. [↑](#footnote-ref-2)