Титульный лист материалов по дисциплине

ДИСЦИПЛИНА **Структуры и алгоритмы обработки данных (ч. 2)**
(полное наименование дисциплины без сокращений)

ИНСТИТУТ **ИТ**

КАФЕДРА **Математического обеспечения и стандартизации информационных технологий**
полное наименование кафедры)

ВИД УЧЕБНОГО МАТЕРИАЛА **Практические работы**
(в соответствии с пп.1-11)

ПРЕПОДАВАТЕЛЬ **Муравьёва Екатерина Андреевна**
(фамилия, имя, отчество)

СЕМЕСТР **3 семестр, 2023-2024 уч. год**
(указать семестр обучения, учебный год)

.

Практическая работа №1
«Поразрядные операции. Сортировка числового файла с помощью битового массива»

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

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

В языке программирования C++ 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).

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

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

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.

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.

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

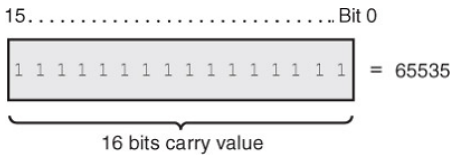

Рис. 1. Беззнаковое двухбайтовое целое число в памяти ЭВМ.

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.

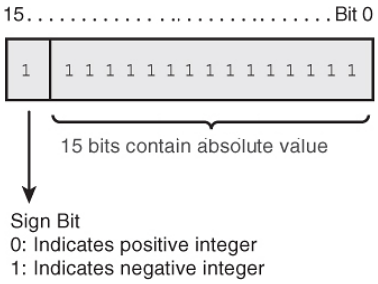

Рис. 2. Двухбайтовое целое число со знаком в памяти ЭВМ.

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

Таблица 2. Битовые операции в C++.

~	Инверсия (0 заменяет на 1, а 1 на 0)	x=0x0F; ~x; результат 0xF0
x<<n	Сдвиг влево двоичного кода (умножение на 2^n)	unsigned int x=7; x=x<<2; результат 0x0000001C
x>>n	Сдвиг вправо двоичного кода (деление на 2^n)	unsigned int x=28; x=x>>2; результат =0x00000007

x & maska	Поразрядное И (применяется для записи в указанный разряд 0)	Правило выполнения операции
		111
		& 100
		= 100
		
		Пример. Установить в двоичном коде переменной x только 9-ый справа бит в 0
		Способы определения маски и решение
		//Способ 1. Маска создана явно
		unsigned short int x1=0xAEFF;
		unsigned short int maska=0xFDFF;
		x1=x1 & maska;
		//Способ 2. Маска формируется инициализируется 1
		maska=1;
		x1=0xFFFF
		x1=x1 & (~ (maska<<9));
		//Способ 3. Маска создается из 1 в выражении
		x1=0xFFFF
		x1=x1 & (~ (1<<9));
		или
		x=x & (~ (1<<9));

X | maska	Поразрядное ИЛИ (применяется для записи в указанный разряд 1)	Правило выполнения операции
		111
		| 100
		111
		Пример. Установить в двоичном коде переменной x 9-ый справа бит в 1.
		Решение.
		unsigned short int x=0xACFF;
		unsigned short int maska=0x0200;
		x=x | maska результат 0xAEFF;
		maska=1;
		x=x | ((maska<<9));
		
		или
		x=x | ((1<<9));

X ^ maska	Исключающее ИЛИ для поразрядных операций. Используется для проверки соответствующих битов двух переменных, если они имеют разные значения, то результат 1, а если равны, то 0.	Правило выполнения операции
		1111
		^ 0001
		= 1110
		unsigned int x=0xF, a=1;
		a=x^a;
		Результат: в переменной ***a*** значение 0x0000000E

Задание 1.

Выполнить упражнения по применению битовых операций по изменению значений битов в ячейке оперативной памяти, созданию маски для изменения значения ячейки.

В таблице 3 приведены варианты индивидуальных заданий. Номер варианта соответствует номеру студента в журнале учета посещений. В каждом варианте 5 упражнений.

Таблица 3. Варианты упражнений.

№	1 Номер бита	2 Номер бита	3 множитель	4 делитель	5 Задание для выражения
1	5-ый и 7-ой справа	С 9-ого четыре бита слева	8	8	Установить n-ый бит в 1, используя маску (вар 1)
2	Три старших	12-ый, 14-ый, 3-ий	4	4	Установить n-ый бит в 1,

					используя маску (вар2)
3	Только с четными номерами	7-ой, 9-ый, 11-ый	16	16	Обнулить n-ый бит, используя маску (вар 1)
4	Только с не четными номерами	С 5-ого бита четыре слева	32	32	Установить n-ый бит в 1, используя маску (вар 2)
5	17-ый, 15-ый, 1-ый	С 5-ого бита три справа	64	64	Обнулить n-ый бит, используя маску (вар 2)
6	3-ий, 11-ый, 5-ый	Четыре младших бита	128	128	Установить n-ый бит в 1, используя маску (вар 2)
7	Четыре старших бита	9-ый, 11-ый, 3-ий	512	512	Обнулить n-ый бит, используя маску (вар 1)
8	1-ый, 6-ой, 9-ый	1-ый, 6-ой, 9-ый	8	8	Установить n-ый бит в 1, используя маску (вар 1)
9	0-ый, 11-ый, 3-ий	Четыре старших бита	4	4	Обнулить n-ый бит, используя маску (вар 2)
10	Четыре младших бита	3-ий, 11-ый, 5-ый	16	16	Установить n-ый бит в 1, используя маску (вар 2)
11	С 5-ого бита четыре слева	5-ый, 7-ой справа	32	32	Обнулить n-ый бит, используя маску (вар 1)
12	С 3-ого бита три справа	Три старших	64	64	Установить n-ый бит в 1, используя маску (вар 1)
13	7-ой, 9-ый, 11-ый	Только с четными номерами	128	128	Обнулить n-ый бит, используя маску (вар 1)
14	12-ый, 14-ый, 3-ий	Только с нечетными номерами	512	512	Установить n-ый бит в 1, используя маску (вар 2)

15	С 9-ого бита четыре слева	17-ий, 15-ый, 1-ый	1024	1024	Обнулить n-ый бит, используя маску (вар 1)
16	1-ый, 2-ой,7-ой	С 7-ого три бита слева	8	8	Обнулить n-ый бит, используя маску (вар 1)
17	С 3-ого бита четыре слева	12-ый, 11-ый, 1-ый	32	32	Обнулить n-ый бит, используя маску (вар 2)
18	3-ий, 5-ый, 8-ой	С 4-ого два бита слева	16	16	Обнулить n-ый бит, используя маску (вар 1)
19	1-ый, 5-ый, 6-ой	15-ый, 12-ый, 3-ий	128	128	Обнулить n-ый бит, используя маску (вар 1)
20	5-ый и 7-ой справа	С 7-ого четыре бита слева	16	8	Установить n-ый бит в 1, используя маску (вар 1)
21	Три младших разряда	11-ый, 12-ый, 3-ий	32	16	Установить n-ый бит в 1, используя маску (вар 2)
22	Только с не четными номерами	5-ой, 3-ый, 11-ый	8	16	Обнулить n-ый бит, используя маску (вар 1)
23	Только с четными номерами	С 15-ого бита четыре справа	16	32	Установить n-ый бит в 1, используя маску (вар 2)
24	18-ый, 22-ый, 1-ый	С 8-ого бита три слева	64	64	Обнулить n-ый бит, используя маску (вар 2)
25	7-ий, 13-ый, 5-ый	Четыре старших бита	512	128	Установить n-ый бит в 1, используя маску (вар 2)
26	Четыре старших бита	9-ый, 11-ый, 3-ий	128	512	Обнулить n-ый бит, используя маску (вар 1)
27	2-ый, 5-ой, 9-ый	2-ый,5-ой, 9-ый	16	32	Установить n-ый бит в 1, используя маску (вар 1)

28	5-ый, 11-ый, 3-ий	Четыре старших бита	32	32	Обнулить n-ый бит, используя маску (вар 2)
29	пять младших битов	7-ий, 9-ый, 5-ый	64	64	Установить n-ый бит в 1, используя маску (вар 2)
30	С 7-ого бита пять слева	5-ый, 7-ой , 11-ый	32	32	Обнулить n-ый бит, используя маску (вар 1)

Разработать программу, которая продемонстрирует выполнение упражнений варианта. Результаты выполнения упражнения выводить на монитор.

Упражнения:

1) Определить переменную целого типа, присвоить ей значение, используя константу в шестнадцатеричной системе счисления.
   Разработать оператор присваивания и его выражение, которое установит заданные в задании биты исходного значения переменной в значение 1, используя соответствующую маску и поразрядную операцию.

2) Определить переменную целого типа.
   Разработать оператор присваивания и его выражение, которое обнуляет заданные в задании биты исходного значения переменной, используя соответствующую маску и поразрядную операцию. Значение в переменную вводится с клавиатуры.

3) Определить переменную целого типа.
   Разработать оператор присваивания и выражение, которое умножает значение переменной на число, указанное в третьем столбце варианта, используя соответствующую поразрядную операцию. Изменяемое число вводится с клавиатуры.

4) Определить переменную целого типа.
   Разработать оператор присваивания и выражение, которое делит значение переменной на число, указанное в четвертом столбце варианта, используя соответствующую поразрядную операцию. Изменяемое число вводится с клавиатуры.

5) Определить переменную целого типа.
   Разработать оператор присваивания и выражение, в котором используются только поразрядные операции. В выражении используется маска – переменная. Маска может быть инициализирована единицей в младшем разряде (вар 1) или единицей в старшем разряде (вар 2). Изменяемое число вводится с клавиатуры.

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.

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***.

Задание 2.

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.

Исправьте программу задания, 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 $n=10^7$ неотрицательных целых чисел, 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.

Реализуйте тестовый пример, демонстрирующий входные данные и заполненный битовый массив (не более 20 чисел).

Реализуйте задачу сортировки числового файла для входных данных объемом 100 и 1000 чисел. Показать время выполнения сортировки для каждого объема.

Реализуйте задачу сортировки заданного числового файла.

В отчёт внесите результаты тестирования для наибольшего количества входных чисел, соответствующего битовому массиву длиной 1МБ и программно определить объём оперативной памяти, занимаемый битовым массивом и время выполнения работы программы (сортировки) для каждого случая.

Содержание отчёта:

1. Титульный лист.
2. Цель работы.
3. Ход работы (по каждому заданию):
    a. Формулировка задачи.
    b. Математическая модель решения (описание алгоритма).
    c. Код программы с комментариями.
    d. 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).

Приложение 1. Пример реализации алгоритма вывода двоичного кода заданного значения

```
void coutp(unsigned int x)
{
    int n=sizeof(int)*8;
    unsigned  maska=(1<<(n-1));
    for(int i=1; i<=n;i++)
    {
            cout<<((x&maska)>>(n-i));
            maska=maska>>1;
    }
}
```