



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Отчет по выполнению практического задания 5.1

**Тема: Битовые операции. Сортировка числового файла с помощью
битового массива**

Дисциплина: Структуры и алгоритмы обработки данных

Выполнил студент:	Зуев Д.А.
Группа:	ИКБО-68-23

Москва 2025

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

Задание 1

Формулировка задачи

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.

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. В числе со знаком под модуль отведены все двоичные разряды, кроме старшего. 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 C++.

1.а Математическая модель решения (описание алгоритма)

Данный алгоритм реализует побитовую операцию для модификации определённого бита в двоичном представлении числа. Переменная `x` инициализируется значением 255, что в двоичной системе соответствует числу `11111111`. Переменной `mask` присваивается значение 1 (`00000001`), которое впоследствии сдвигается влево на 4 позиции, преобразуясь в двоичное число `00010000`. Затем происходит инвертирование маски, в результате чего она принимает значение `11101111`, что эквивалентно сбросу пятого бита. Далее выполняется побитовая операция И между переменной `x` и данной маской, что приводит к обнулению пятого бита в числе `x`. Итоговое значение переменной `x` составляет 239, что соответствует результату операции и выводится на экран.

1.а Код программы

Реализуем код этой программы на языке C++. Результат представлен на рисунке ниже (рис. 1.1).

```
1563 //Номер 1a
1564 #include <iostream>
1565 int main()
1566 {
1567     unsigned x = 255;
1568     unsigned maska = 1;
1569     x = x & (~(maska << 4));
1570     std::cout << x;
1571 }
```

Рисунок 1.1 – Реализация кода задачи

1.а Результаты тестирования

Запустим данный код без входных данных поскольку эта задача не требует. Результат представлен ниже (рис. 1.2).

239

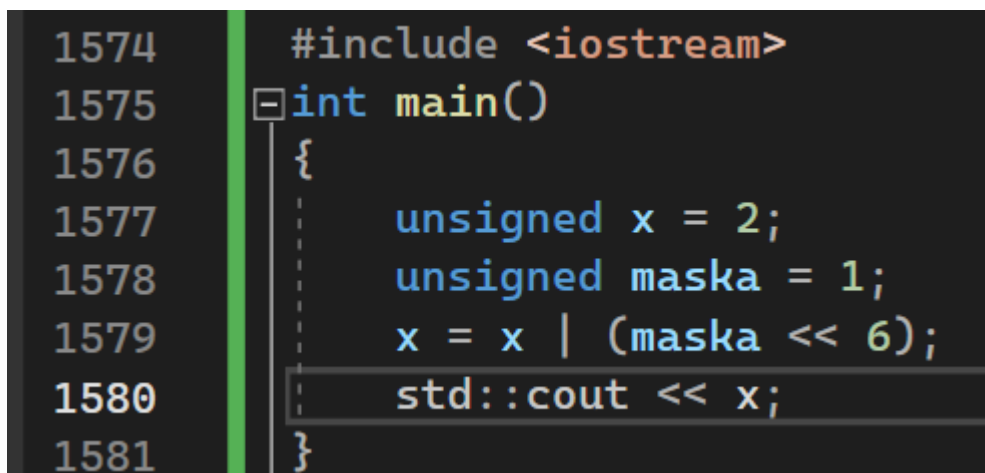
Рисунок 1.2 – Реализация кода задачи

1.6 Математическая модель решения (описание алгоритма)

Этот код выполняет установку 7-го бита числа в единицу. Сначала переменной `x` присваивается значение 2, что в двоичном представлении равно `00000010`. Затем переменной `mask` присваивается значение 1, представленное как `00000001`. Операцией сдвига влево на 6 позиций маска преобразуется в значение `01000000`, что соответствует установке 7-го бита в единицу. В дальнейшем выполняется побитовая операция ИЛИ между переменной `x` и маской, что устанавливает 7-й бит переменной `x` в единицу. Результат операции — значение 66 (`01000010`), которое выводится на экран.

1.6 Код программы

Реализуем код этой программы на языке C++. Результат представлен на рисунке ниже (рис. 2.1).



```
1574 #include <iostream>
1575 int main()
1576 {
1577     unsigned x = 2;
1578     unsigned maska = 1;
1579     x = x | (maska << 6);
1580     std::cout << x;
1581 }
```

Рисунок 2.1 – Реализация кода задачи

1.6 Результаты тестирования

Запустим данный код без входных данных поскольку эта задача не требует. Результат представлен ниже (рис. 2.2).



```
66
```

Рисунок 2.2 – Реализация кода задачи

1.в Математическая модель решения (описание алгоритма)

Данный код реализует побитовый вывод числа с использованием маски для итеративной проверки каждого бита, начиная с самого старшего.

переменной и проверяет, какие биты установлены (то есть, какие числа были введены). В итоге выводятся уникальные числа в порядке возрастания, на основе установленных битов.

2.a Код программы

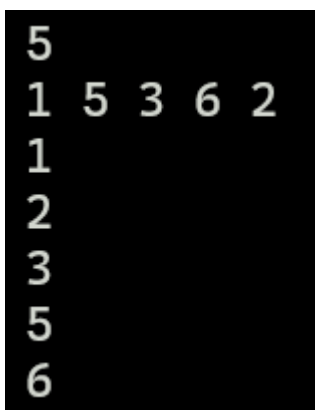
Реализуем код этой программы на языке C++. Результат представлен ниже.

```
1595 //Номер 2a
1596 #include <iostream>
1597 #include <bitset>
1598 int main()
1599 {
1600     int n;
1601     std::cin >> n;
1602     while (n > 8 || n < 1)
1603     {
1604         std::cin >> n;
1605     }
1606
1607     unsigned char l = 0, maska = 1;
1608     for (int i = 0; i < n; i++)
1609     {
1610         int x;
1611         std::cin >> x;
1612         l = l | (maska << x);
1613     }
1614
1615     for (int i = 0; i < 8; i++)
1616     {
1617         if (l & (maska << i))
1618         {
1619             std::cout << i << "\n";
1620         }
1621     }
1622 }
```

2.a Результаты тестирования

Запустим данный код с последовательностью длины 5 вида {1, 5, 3, 6, 2}.

Результат работы представлен на Рисунке 4.1



```
5
1 5 3 6 2
1
2
3
5
6
```

Рисунок 4.1 – Реализация кода задачи

Как можно увидеть, последовательность была в правильном отсортированном порядке.

2.6 Математическая модель решения (описание алгоритма)

Программа начинает с того, что запрашивает у пользователя количество чисел, которые он хочет ввести, при этом количество должно быть в диапазоне от 1 до 64. Если ввод некорректен, программа повторяет запрос, пока не будет введено корректное значение.

Затем программа последовательно запрашивает сами числа, которые должны находиться в диапазоне от 0 до 63. Каждое число сохраняется в виде битовой маски, где позиция установленного бита соответствует введённому числу. Для этого используется операция побитового сдвига и ИЛИ, которая устанавливает нужный бит в переменной. После ввода всех чисел программа проверяет, какие биты в переменной установлены (то есть какие числа были введены). Она выводит эти числа в порядке возрастания на основе позиций установленных битов, что фактически представляет собой отсортированный вывод введённых чисел.

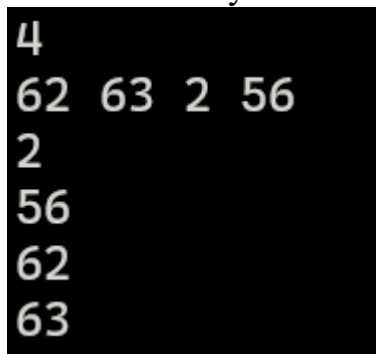
2.6 Код программы

Реализуем код этой программы на языке C++. Результат представлен ниже.

```
1624 //Номер 26
1625 #include <iostream>
1626 int main()
1627 {
1628     int n;
1629     std::cin >> n;
1630     while (n > 64 || n < 1)
1631     {
1632         std::cin >> n;
1633     }
1634
1635     unsigned long long int l = 0, maska =
1636     for (int i = 0; i < n; i++)
1637     {
1638         int x;
1639         std::cin >> x;
1640         l = l | (maska << x);
1641     }
1642
1643     for (int i = 0; i < 64; i++)
1644     {
1645         if (l & (maska << i))
1646         {
1647             std::cout << i << "\n";
1648         }
1649     }
1650 }
```

2.6 Результаты тестирования

Запустим данный код с последовательностью длины 4 вида {62, 63, 2, 56,}. Результат работы представлен на Рисунке 5.1



```
4
62 63 2 56
2
56
62
63
```

Рисунок 5.1 – Реализация кода задачи

Как можно увидеть, последовательность была в правильном отсортированном порядке.

2.в Математическая модель решения (описание алгоритма)

Программа начинается с ввода количества чисел, которые будут записаны в массив, с проверкой, что это число находится в диапазоне от 1 до 64. Если введено некорректное значение, программа продолжает запрашивать ввод, пока не будет введено корректное.

Затем создаётся массив из 8 элементов типа `'unsigned char'`, каждый из которых представляет собой 8 бит (итого 64 бита), чтобы хранить числа в диапазоне от 0 до 63. Пользователь вводит числа одно за другим, и программа сохраняет эти числа в массив, используя побитовые операции. Для каждого числа определяется соответствующий элемент массива и устанавливается бит, соответствующий номеру введённого числа.

После ввода всех чисел программа проверяет массив, определяя, какие биты установлены. Для каждого установленного бита выводится его индекс, что соответствует введённому числу. Таким образом, программа выводит отсортированный список уникальных чисел, введённых пользователем.

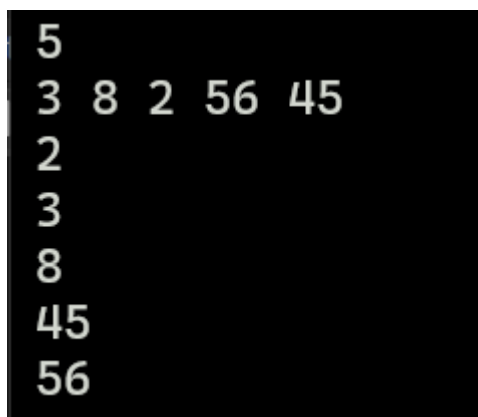
2.в Код программы

Реализуем код этой программы на языке C++. Результат представлен ниже.

```
1652 //Номер 2в
1653 #include <iostream>
1654 #include <vector>
1655 int main()
1656 {
1657     int n;
1658     std::cin >> n;
1659     while (n > 64 || n < 1)
1660     {
1661         std::cin >> n;
1662     }
1663     unsigned char m[8] = { 0, 0, 0, 0, 0, 0, 0, 0 };
1664     unsigned long long int maska = 1;
1665     for (int i = 0; i < n; i++)
1666     {
1667         int x;
1668         std::cin >> x;
1669         m[x / 8] = m[x / 8] | (maska << (x % 8));
1670     }
1671
1672     for (int i = 0; i < 64; i++)
1673     {
1674         if (m[i / 8] & (maska << (i % 8)))
1675         {
1676             std::cout << i << "\n";
1677         }
1678     }
1679 }
```

2.в Результаты тестирования

Запустим данный код с последовательностью длины 5 вида {3, 8, 2, 56, 45} Результат работы представлен на Рисунке 6.1



```
5
3 8 2 56 45
2
3
8
45
56
```

Рисунок 6.1 – Реализация кода задачи

Как можно увидеть, последовательность была в правильном отсортированном порядке.

Задание 3

Формулировка задачи

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.

3 Математическая модель решения (описание алгоритма)

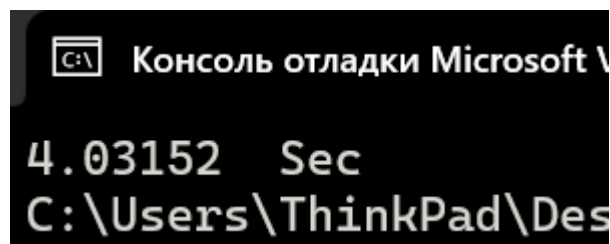
Программа начинается с открытия файла, в котором хранятся неотсортированные числа. Затем она создаёт структуру данных, состоящую из битовых масок, где каждый бит будет соответствовать одному числу в заданном диапазоне. Программа читает каждое число из входного файла и устанавливает соответствующий бит в созданной структуре. После обработки всех чисел она открывает новый файл для записи отсортированных чисел. Программа проходит по битовой структуре, проверяет, какие биты установлены, и записывает соответствующие числа в выходной файл в отсортированном порядке. В завершение программа измеряет время, затраченное на выполнение задачи, и рассчитывает объём памяти, использованный для хранения битовой структуры, выводя обе эти информации на экран

1.а Код программы

Реализуем код этой программы на языке C++. Результат представлен ниже.

```
1681 //Номер 3а и 3б
1682 #include <iostream>
1683 #include <fstream>
1684 #include <chrono>
1685 #include <vector>
1686 int main()
1687 {
1688     std::ofstream file("text.txt");
1689     for (auto i = 999999; i > -1; i--)
1690     {
1691         file << i << "\n"; //Заполнение файла;
1692     }
1693     file.close();
1694     auto start = std::chrono::high_resolution_clock::now();
1695     std::vector<unsigned int> l(10000000);
1696     std::ifstream file_read("text.txt");
1697     std::string tmp;
1698
1699     while (std::getline(file_read, tmp))
1700     {
1701         unsigned int i = std::stoi(tmp);
1702         l[i] = 1;
1703     }
1704
1705     std::ofstream file_write("text.txt");
1706     for (auto i = 0; i < 10000000; i++)
1707     {
1708         if (l[i] == 1)
1709         {
1710             file_write << i << "\n";
1711         }
1712     }
1713     file_write.close();
1714     auto end = std::chrono::high_resolution_clock::now();
1715     std::chrono::duration<double> time = end - start;
1716     std::cout << time.count() << " Sec";
1717     return 0;
1718 }
```

Запустим данный код (рис. 7.1)



Последовательность отсортирована правильно, значит программа работает верно.

Выполнение данного кода затратило следующие ресурсы (рис. 7.2)

Рисунок 7.2 – Затраченные ресурсы

```
Объем оперативной памяти, занимаемый битовым массивом: 1048576 байт (1 МБ)  
Размер указателя: 8 байт
```

ВЫВОДЫ

В рамках выполненной практической работы цель по освоению приёмов работы с битовым представлением беззнаковых целых чисел и реализации эффективного алгоритма внешней сортировки на основе битового массива была успешно достигнута. Написанные программы продемонстрировали умелое использование битового представления для хранения и обработки больших объёмов данных. Реализованный алгоритм внешней сортировки с использованием битового массива показал высокую эффективность в обработке и сортировке чисел. Были освоены работы с битовым представлением беззнаковых целых чисел.

ЛИТЕРАТУРА

1. Страуструп Б. Программирование. Принципы и практика с использованием C++. 2-е изд., 2016.
2. Документация по языку C++ [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)