|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  | |
|  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 2.1** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Сортировка числового файла с помощью битового массива»** | |
|  | |
| Выполнил студент группы ИКБО-30-22 | Сенькевич Г.Д. |
| Принял преподаватель | Красников С.А. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2023

# **Цель работы**

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

1. **Ход работы**

# **. Формулировка задачи**

**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.

**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.

* 1. **. Математическая модель решения**

Для решения задачи были использованы такие операции как битовый сдвиг и побитовые логические операции.

1. Битовый сдвиг:

* Левый битовый сдвиг (<<): Эта операция сдвигает биты числа влево на указанное количество позиций.
* Правый битовый сдвиг (>>): Эта операция сдвигает биты числа вправо на указанное количество позиций.

1. Побитовые логические операции:

* Побитовое И (&): Эта операция выполняет логическое И между каждой парой битов двух чисел. Результат равен 1, только если оба бита равны 1.
* Побитовое ИЛИ (|): Эта операция выполняет логическое ИЛИ между каждой парой битов двух чисел. Результат равен 1, если хотя бы один из битов равен 1.

С помощью этих операций программист получает возможность взаимодействовать с отдельными битами чисел, что позволяет реализовать сортировку с использованием битового массива.

Сортировка с использованием битового массива — это метод, который позволяет упорядочить набор элементов путем использования битов для отображения присутствия или отсутствия каждого элемента в сортированном массиве. Вот как это происходит:

1. **Инициализация битового массива:** создается битовый массив, который будет использоваться для отслеживания присутствия элементов в исходном массиве. Размер битового массива определяется максимальным значением элементов в исходном массиве.
2. **Установка битов:** для каждого элемента в исходном массиве происходит установка соответствующего бита в битовом массиве. Например, если элемент в исходном массиве равен 5, то бит с индексом 5 в битовом массиве устанавливается в 1, что указывает на то, что элемент присутствует в исходном массиве.
3. **Извлечение элементов:** после того как все элементы из исходного массива были обработаны и битовый массив правильно настроен, можно начать извлечение элементов в сортированном порядке. Для этого проходят по битовому массиву и извлекают элементы, соответствующие установленным битам. Эти элементы извлекаются в порядке возрастания индексов битов.
4. **Сортированный массив:** получается отсортированный массив, в котором элементы упорядочены по возрастанию.
   1. **. Код программы**

Решение задания 1.а. приведено в листинге 1. Тестирование на других значениях x приводится в разделе 2.4.

Листинг 1 ­— Решение задания 1.а.

|  |
| --- |
| #include <iostream>  int main()  {  unsigned char x = 255;  unsigned char mask = 1;  x = x & (~(mask << 4));  std::cout << (int)x;  return 0;  } |

Решение задания 1.б. приведено в листинге 2. Для того чтобы установить 7 бит числа в единицу нам нужно сдвинуть маску (изначально равную 1) на 6 позиций влево и провести с ней побитовое «или». Таким образом 7 биту присвоится значение операции «или» от 1 и значения, которое там находилось, что в любом случае даст 1.

Листинг 2 ­— Решение задания 1.б.

|  |
| --- |
| #include <iostream>  int main()  {  unsigned char x = 0;  unsigned char mask = 1;  x = x | (mask << 6);  std::cout << (int)x;  return 0;  } |

Решение задания 1.в. приведено в листинге 3. Этот код выводит биты числа 25 с помощью маски. Единичный бит маски последовательно сдвигается из крайней левой позиции направо, и на каждой итерации цикла с помощью маски происходит получение очередного бита числа x и вывод его на экран.

Листинг 3 ­— Решение задания 1.в.

|  |
| --- |
| #include <cstdlib>  #include <iostream>  #include <Windows.h>  #include <bitset>  using namespace std;  int main()  {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  unsigned int x = 25;  const int n = sizeof(int) \* 8;  unsigned maska = (1 << n - 1);  cout << "Начальный вид маски: " << bitset<n> (maska) << endl;  cout << "Результат: ";  for (int i = 1; i <= n; i++)  {  cout << ((x & maska) >> (n - i));  maska = maska >> 1;  }  cout << endl;  system("pause");  return 0;  } |

Решение задания 2.а. приведено в листинге 4. В первом цикле происходит последовательное считывание чисел и запись их в битовый массив (запись единиц на позиции с номером считанных чисел). Во втором цикле происходит последовательная проверка битов битового массива и вывод позиций единичных битов. Таким образом на экран выводится отсортированный массив.

Листинг 4 ­— Решение задания 2.а.

|  |
| --- |
| #include <iostream>  //#include <bitset>  typedef unsigned char byte;  int main()  {  const size\_t size = 8;  byte bit\_array = 0;  for (int i = 0; i < size; i++)  {  int buff = 0;  std::cin >> buff;  byte mask = 1;  bit\_array = bit\_array | (mask << buff);  }  //std::cout << std::bitset<size> (bit\_array);  byte mask = 1;  for (int i = 0; i < size; i++)  {  if (bit\_array & (mask << i))  {  std::cout << i << ' ';  }  }  std::cout << std::endl;    return 0;  }0;  } |

Решение задачи 2.б. приведено в листинге 5. Решение аналогично решению задания 2.а., с отличием в размере массива и границах допустимых значений, а соответственно и типе переменной, используемой для представления битового массива (64 бита вместо 8).

Листинг 5 ­— Решение задания 2.б.

|  |
| --- |
| #include <iostream>  //#include <bitset>  typedef unsigned long long ull;  int main()  {  const size\_t size = 64;  ull bit\_array = 0;  for (int i = 0; i < size; i++)  {  ull buff = 0;  std::cin >> buff;  ull mask = 1;  bit\_array = bit\_array | (mask << buff);  }  //std::cout << std::bitset<size> (bit\_array);  ull mask = 1;  for (int i = 0; i < size; i++)  {  if (bit\_array & (mask << i))  {  std::cout << i << ' ';  }  }  std::cout << std::endl;  return 0;  } |

Решение задания 2.в. приведено в листинге 6. Решение аналогично решениям заданий 2.а. и 2.б. Отличие заключается в том что битовый массив реализуется не через одну числовую переменную, а через массив 1-байтовых числовых переменных, каждая из которых представляет собой отдельный битовый массив. Это создаёт накладные расходы для записи числа в массив, так как сначала нужно вычислить индекс элемента, в который нужно записать число, а только потом позицию бита, в который нужно установить 1. Последующий вывод чисел из массива также сопровождается дополнительными вычислениями.

Листинг 6 ­— Решение задания 2.в.

|  |
| --- |
| #include <iostream>  typedef unsigned long long ull;  typedef unsigned char byte;  int main()  {  const size\_t size = 64;  const size\_t byte\_size = sizeof(byte) \* 8;  const size\_t byte\_array\_size = size / byte\_size;  byte\* byte\_array = new byte[byte\_array\_size];  for (int i = 0; i < byte\_array\_size; i++)  {  byte\_array[i] = 0;  }  for (int i = 0; i < size; i++)  {  ull buff = 0;  std::cin >> buff;  size\_t index = buff / byte\_size;  byte shift = buff % byte\_size;  byte mask = 1;  byte\_array[index] = byte\_array[index] | (mask << shift);  }  byte mask = 1;  for (int i = 0; i < byte\_array\_size; i++)  {  for (int j = 0; j < byte\_size; j++)  {  if (byte\_array[i] & (mask << j))  {  std::cout << i \* byte\_size + j << ' ';  }  }  }  std::cout << std::endl;  delete[] byte\_array;  return 0;  } |

Решение задач 3.а. и 3.б. приведено в листинге 7. Решение аналогично решению задания 2.в. с отличием в том что размер массива вместо 64 бит составляет 1 Мб, а чтение и вывод информации происходят с помощью файловых потоков, а не стандартных. Также замеряется время, затрачиваемое на чтение информации из файла, записи этой информации в битовый массив (что одновременно является и сортировкой этих данных) и записи отсортированного массива в выходной файл.

Листинг 7 ­— Решение заданий 3.а. и 3.б.

|  |
| --- |
| #include <iostream>  #include <fstream>  #include <string>  #include <chrono>  typedef unsigned long long ull;  typedef unsigned char byte;  int main()  {  const size\_t size = 8388608; // 1 Mb in bits  const size\_t byte\_size = sizeof(byte) \* 8;  const size\_t byte\_array\_size = size / byte\_size;  byte\* byte\_array = new byte[byte\_array\_size];  for (int i = 0; i < byte\_array\_size; i++)  {  byte\_array[i] = 0;  }  std::cout << "Enter filename: ";  std::string filename;  std::cin >> filename;  std::ifstream input\_stream(filename);  auto start = std::chrono::high\_resolution\_clock::now();  ull buff = 0;  while (input\_stream >> buff)  {  size\_t index = buff / byte\_size;  byte shift = buff % byte\_size;  byte mask = 1;  byte\_array[index] = byte\_array[index] | (mask << shift);  }  std::ofstream output\_stream("out.txt");  byte mask = 1;  for (int i = 0; i < byte\_array\_size; i++)  {  for (int j = 0; j < byte\_size; j++)  {  if (byte\_array[i] & (mask << j))  {  output\_stream << i \* byte\_size + j << ' ';  }  }  }  auto stop = std::chrono::high\_resolution\_clock::now();  auto duration = std::chrono::duration\_cast<std::chrono::milliseconds>(stop - start);  std::cout << "Reading, sorting and writing took " << duration.count() / 1000.0 << " seconds\n";  delete[] byte\_array;  return 0;  } |

* 1. **. Результаты тестирования**

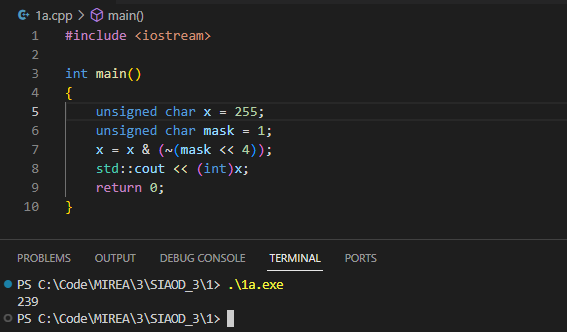
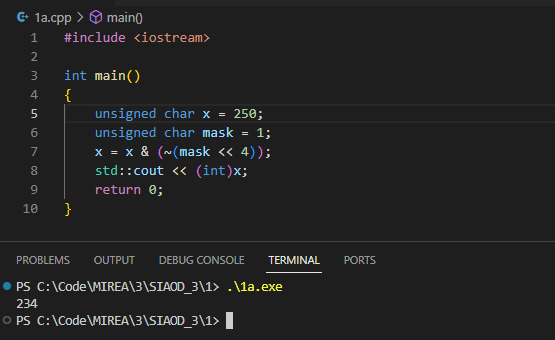
Результаты тестирования решения задания 1.а. представлены на рисунках 1 и 2.

Рисунок 1 — Тестирование решения задания 1.а.

Рисунок 2 — Тестирование решения задания 1.а.

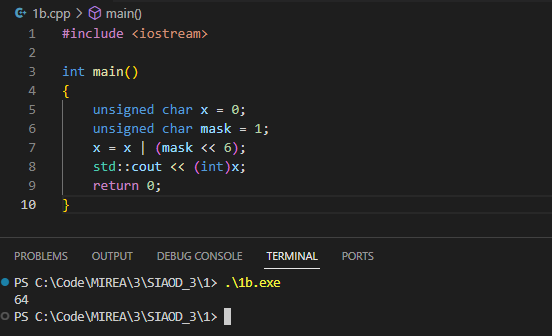
 Результаты тестирования решения задания 1.б. представлены на рисунке 3.

Рисунок 3 — Тестирование решения задания 1.б.

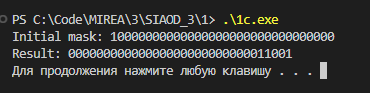
Результаты тестирования решения задания 1.в. представлены на рисунке 4.

Рисунок 4 — Тестирование решения задания 1.в.

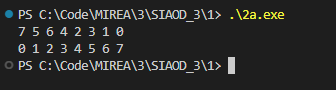
Результаты тестирования решения задания 2.а. представлены на рисунке 5.

Рисунок 5 — Тестирование решения задания 2.а.

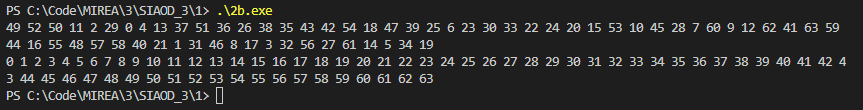
Результаты тестирования решения задания 2.б. представлены на рисунке 6.

Рисунок 6 — Тестирование решения задания 2.б.

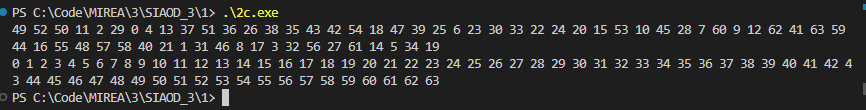
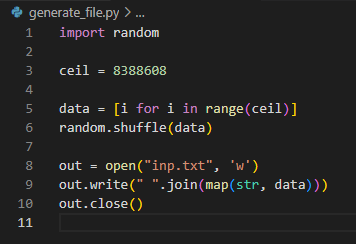
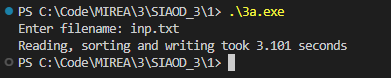
Результаты тестирования решения задания 2.в. представлены на рисунке 7.

Рисунок 7 — Тестирование решения задания 2.в.

Для тестирования решения заданий 3.а. и 3.б. был написан скрипт, генерирующий входной файл, состоящий из 8388608 (это количество бит в одном мегабайте) чисел в диапазоне от 0 до 8388608, записанных в случайном порядке. Скрипт представлен на рисунке 8. Результаты тестирования программы представлены на рисунках 9 и 10. Выводимое время — это время, затраченное на чтение данных из файла, запись их в битовый массив и последующий вывод из массива в выходной файл.

Рисунок 8 — Скрипт, генерирующий входные данные для заданий 3.а. и 3.б.



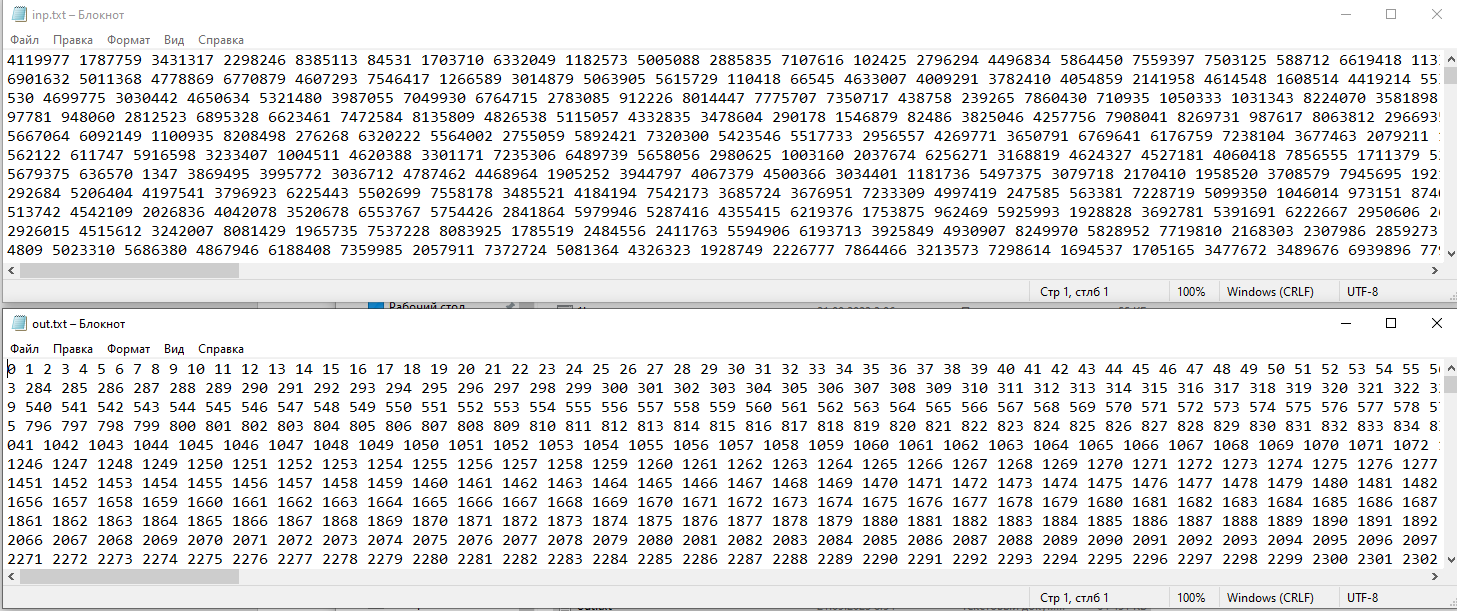
Рисунок 9 — Тестирование решения заданий 3.а. и 3.б.

Рисунок 9 — Тестирование решения заданий 3.а. и 3.б.

Тестирование показало, что все программы работают правильно, корректно решая поставленные задачи.

1. **Выводы**

В результате выполнения работы я:

1. Освоил приёмы работы с битовым представлением беззнаковых целых чисел;
2. Реализовать эффективный алгоритм внешней сортировки на основе битового массива.
3. **Список литературы**
4. Страуструп Б. Программирование. Принципы и практика с использованием C++. 2-е изд., 2016.
5. Документация по языку С++ [Электронный ресурс]. URL: https://docs.microsoft.com/ruru/cpp/cpp/ (дата обращения 01.09.2021).
6. 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).