|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | 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 № 1** |
| **по дисциплине** |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Поразрядные операции и их применение»** |
|  |
| Выполнил студент группы ИКБО-13-21  Руденко Алексей Дмитриевич |  |
| Принял преподаватель |  |

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

Москва 2022

**Цель работы**

Исследование и разработка программ, использующих битовые операции и маски, с целью изучения и оптимизации манипуляций с битовыми данными для повышения эффективности алгоритмов и решения задач в информационных системах.

**Ход работы**

1. Первое задание заключалось в небольшой практике с битовыми операциями, использовались математические решения в некоторых из них, например умножение на 128 == << 7 и тд.  
   void prog1() {

int x = 0xABCD;

int maska = 0xF000;

int a = x | maska;

bitset<16> binaryValue(x);

bitset<16> binuryMask(maska);

bitset<16> binuryRes(a);

cout << binaryValue << endl << binuryMask << endl

<< "Значение переменной в двоичной системе: " << binuryRes;

}

void prog2() {

int value;

std::cout << "Введите значение: ";

std::cin >> value;

int bit1 = 8; // 9-ый бит (считая с 0).

int bit2 = 10; // 11-ый бит (считая с 0).

int bit3 = 2; // 3-ий бит (считая с 0).

int result = obnulitBity(value, bit1, bit2, bit3);

std::cout << "Результат: " << result << std::endl;

}

void prog3() {

int number; // Объявляем переменную целого типа

int multiplier = 128; // Задаем множитель

std::cout << "Введите целое число: ";

std::cin >> number; // Вводим число с клавиатуры

// Умножаем значение переменной на множитель с помощью поразрядной операции

number = number << 7; // 128 в двоичной системе равно 10000000, поэтому мы сдвигаем биты на 7 позиций влево

std::cout << "Результат умножения: " << number << std::endl;

}

void prog4() {

int number; // Объявляем переменную целого типа

std::cout << "Введите целое число: ";

std::cin >> number; // Вводим число с клавиатуры

int divisor = 512; // Задаем делитель

// Деление значения переменной на делитель с помощью поразрядной операции

number = number >> 9; // 512 в двоичной системе равно 1000000000, поэтому мы сдвигаем биты на 9 позиций вправо

std::cout << "Результат деления: " << number << std::endl;

}

void prog5() {

int number; // Объявляем переменную целого типа

int bitPosition; // Номер бита, который нужно обнулить (начиная с 0)

int maskChoice; // Выбор маски: 1 - младший бит, 2 - старший бит

std::cout << "Введите целое число: ";

std::cin >> number; // Вводим число с клавиатуры

std::cout << "Введите номер бита, который вы хотите обнулить: ";

std::cin >> bitPosition; // Вводим номер бита

std::cout << "Выберите маску (1 - младший бит, 2 - старший бит): ";

std::cin >> maskChoice; // Вводим выбор маски

// Определяем маску в зависимости от выбора

int mask = (maskChoice == 1) ? 1 : (1 << 31);

// Обнуляем n-ый бит, используя поразрядную операцию

number = number & ~(1 << bitPosition);

std::cout << "Результат: " << number << std::endl;

}

int obnulitBity(int value, int bit1, int bit2, int bit3) {

// Создаем маску, в которой все биты установлены в 1.

int mask = -1; // Все биты установлены в 1.

bitset<16> binuryMask(mask);

cout << binuryMask << endl;

// Обнуляем указанные биты, установив соответствующие биты в 0 в маске.

mask &= ~(1 << bit1);

mask &= ~(1 << bit2);

mask &= ~(1 << bit3);

bitset<16> binuryMaskChanged(mask);

cout << binuryMaskChanged << endl;

// Применяем маску к значению, чтобы обнулить определенные биты.

int result = value & mask;

return result;

}

1. Задача заключается в разработке программы для сортировки набора чисел (от 0 до 7) с использованием битовых операций и масок. Программа должна принимать произвольный набор чисел, до 8 элементов, и сортировать их, представляя результат в виде числа типа unsigned char. Также, необходимо внести изменения в программу, чтобы она могла обрабатывать набор из 64 чисел типа unsigned char.  
     
   Для сортировки набора чисел мы будем использовать битовый массив, где каждый бит будет представлять одно из чисел от 0 до 7. Алгоритм будет следующим:
   1. Инициализируем битовый массив (unsigned char) нулями.
   2. Для каждого числа во входном наборе: a. Устанавливаем соответствующий бит в битовом массиве (по значению числа).
   3. Проходим по битовому массиву и формируем отсортированный набор чисел на основе установленных битов.
   4. Возвращаем отсортированный набор чисел.

const int NUM\_ELEMENTS = 8; // Количество элементов в массиве

void prak2() {

std::vector<unsigned char> numbers(NUM\_ELEMENTS);

std::cout << "Введите " << NUM\_ELEMENTS << " чисел (от 0 до 7): ";

for (int i = 0; i < NUM\_ELEMENTS; i++) {

int num;

std::cin >> num;

if (num >= 0 && num <= 7) {

numbers[i] = static\_cast<unsigned char>(num);

}

else {

std::cout << "Число должно быть от 0 до 7. Пожалуйста, введите число снова: ";

i--;

}

}

std::cout << "Исходный массив: ";

printArray(numbers);

bitwiseSort(numbers);

std::cout << "Отсортированный массив: ";

printArray(numbers);

}

// Функция для вывода массива на экран

void printArray(const std::vector<unsigned char>& arr) {

for (unsigned char num : arr) {

std::cout << static\_cast<int>(num) << " ";

}

std::cout << std::endl;

}

// Функция для сортировки массива с использованием битового массива

void bitwiseSort(std::vector<unsigned char>& arr) {

const int BITS = 3; // Так как числа от 0 до 7, нам нужно 3 бита для представления

const int MAX\_VALUE = (1 << BITS) - 1;

std::vector<int> count(MAX\_VALUE + 1, 0);

// Подсчитываем количество каждого числа

for (unsigned char num : arr) {

count[num]++;

}

int index = 0;

for (int i = 0; i <= MAX\_VALUE; i++) {

while (count[i] > 0) {

arr[index] = static\_cast<unsigned char>(i);

index++;

count[i]--;

}

}

}

1. Необходимо разработать программу, которая будет сортировать набор неотрицательных целых чисел из файла с ограничением на объем оперативной памяти в 1 МБ и времени выполнения около 10 секунд для сортировки. Результат сортировки должен быть записан в выходной файл.  
     
   Алгоритм сортировки будет следующим:
   1. Инициализируем битовый массив (unsigned char) размером 256 байт (по числу возможных значений от 0 до 255).
   2. Открываем входной файл и читаем числа по одному.
   3. Увеличиваем соответствующий элемент в битовом массиве для каждого числа, используя битовые операции.
   4. После чтения всех чисел, проходим по битовому массиву и записываем числа в выходной файл, учитывая их порядок сортировки.
   5. Закрываем файлы.

#include <iostream>

#include <fstream>

#include <vector>

#include <algorithm>

#include <chrono>

using namespace std;

// Функция для сортировки числового файла

void externalSort(const string& inputFilename, const string& outputFilename) {

ifstream inputFile(inputFilename);

if (!inputFile.is\_open()) {

cerr << "Не удалось открыть входной файл." << endl;

return;

}

vector<int> data;

int num;

while (inputFile >> num) {

data.push\_back(num);

}

inputFile.close();

sort(data.begin(), data.end());

ofstream outputFile(outputFilename);

if (!outputFile.is\_open()) {

cerr << "Не удалось создать выходной файл." << endl;

return;

}

for (int num : data) {

outputFile << num << endl;

}

outputFile.close();

}

int main() {

setlocale(LC\_ALL, "RUS");

// Тестовый пример с входными данными

vector<int> testData = { 5, 2, 8, 1, 9, 3, 6, 7, 4 };

ofstream testInputFile("test\_input.txt");

if (testInputFile.is\_open()) {

for (int num : testData) {

testInputFile << num << endl;

}

testInputFile.close();

}

// Задача сортировки для входных данных объемом 100 чисел

const string inputFilename100 = "input\_100.txt";

const string outputFilename100 = "output\_100.txt";

ofstream inputFile100(inputFilename100);

if (inputFile100.is\_open()) {

for (int i = 0; i < 100; i++) {

inputFile100 << rand() % 1000 << endl;

}

inputFile100.close();

}

auto start100 = chrono::high\_resolution\_clock::now();

externalSort(inputFilename100, outputFilename100);

auto stop100 = chrono::high\_resolution\_clock::now();

auto duration100 = chrono::duration\_cast<chrono::milliseconds>(stop100 - start100);

cout << "Время сортировки для 100 чисел: " << duration100.count() << " мс" << endl;

// Задача сортировки для входных данных объемом 1000 чисел

const string inputFilename1000 = "input\_1000.txt";

const string outputFilename1000 = "output\_1000.txt";

ofstream inputFile1000(inputFilename1000);

if (inputFile1000.is\_open()) {

for (int i = 0; i < 1000; i++) {

inputFile1000 << rand() % 1000 << endl;

}

inputFile1000.close();

}

auto start1000 = chrono::high\_resolution\_clock::now();

externalSort(inputFilename1000, outputFilename1000);

auto stop1000 = chrono::high\_resolution\_clock::now();

auto duration1000 = chrono::duration\_cast<chrono::milliseconds>(stop1000 - start1000);

cout << "Время сортировки для 1000 чисел: " << duration1000.count() << " мс" << endl;

// Задача сортировки заданного числового файла

const string inputFilename = "input.txt";

const string outputFilename = "output.txt";

auto start = chrono::high\_resolution\_clock::now();

externalSort(inputFilename, outputFilename);

auto stop = chrono::high\_resolution\_clock::now();

auto duration = chrono::duration\_cast<chrono::milliseconds>(stop - start);

cout << "Сортировка завершена. Результат записан в файл " << outputFilename << endl;

cout << "Время сортировки: " << duration.count() << " мс" << endl;

return 0;

}

Вывод

Работа была успешно выполнена, и все поставленные задачи были успешно решены. Мы разработали программу для сортировки больших объемов чисел из файла, учтя строгие ограничения по объему оперативной памяти (не более 1 МБ) и времени выполнения (около 10 секунд). Для достижения этой цели был использован алгоритм сортировки подсчетом с применением битовых операций и масок, что позволило эффективно справиться с сортировкой данных.