МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

"МИРЭА - Российский технологический университет"

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №5.1

по дисциплине
«Структуры и алгоритмы обработки данных»

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

Выполнил студент группы группа ИКБО-10-23 Лазаренко С.А.

Приняла доцент кафедры Макеева О.В.

Москва 2024

ОГЛАВЛЕНИЕ

ЦЕЛЬ РАБОТЫ 3
ЗАДАНИЕ №1 4
1.1 Формулировка задачи 1.а 4
1.2 Описание математической модели 4
1.3 Реализация алгоритма на языке C++ 4
1.4 Формулировка задачи 1.б 4
1.5 Описание математической модели 5
1.6 Реализация алгоритма на языке C++ 5
1.7 Формулировка задачи 1.в 5
1.8 Описание математической модели 5
1.9 Реализация алгоритма на языке C++ 5
ЗАДАНИЕ №2 7
2.1 Формулировка задачи 2.а 7
2.2 Описание математической модели 7
2.3 Реализация алгоритма на языке C++ 7
2.4 Формулировка задачи 2.б 8
2.5 Описание математической модели 8
2.6 Реализация алгоритма на языке C++ 8
2.7 Формулировка задачи 2.в 9
2.8 Описание математической модели 9
2.9 Реализация алгоритма на языке C++ 9
ЗАДАНИЕ №3 11
3.1 Формулировка задачи 3.а-3.б 11
3.2 Описание математической модели 11
3.3 Реализация алгоритма на языке C++ 11
ВЫВОДЫ 14
ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ 15

ЦЕЛЬ РАБОТЫ

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

Сделать выводы о проделанной работе, основанные на полученных результатах.

ЗАДАНИЕ №1

1.1 Формулировка задачи 1.а

Реализовать пример, оформленный в задании, проверить правильность результата в том числе и на других значениях x.

1.2 Описание математической модели

Исходное число представлено в двоичной форме, и с помощью сдвига маски влево и её инверсии создаётся шаблон, который обнуляет нужный бит числа. В данном случае, число 255 (11111111 в двоичной системе) подвергается операции побитового И с инверсией маски, сдвинутой на 4 бита, что приводит к обнулению 5-го бита, результатом чего становится число 239 (11101111 в двоичной системе).

1.3 Реализация алгоритма на языке C++

Рассмотрим код:

```cpp
#include <iostream>
using namespace std;

int main() {
  setlocale(LC_ALL, "RUS");
  unsigned x = 255; // 8-разрядное двоичное число 11111111
  unsigned maska = 1; // 1=00000001 - 8-разрядная маяка
  cout << "До преобразований: " << x << endl;
  x = x & (~(maska << 4));
  cout << "После преобразований: " << x << endl;
}
```

Рисунок 1 – Реализация примера

Результат работы программы представлен на рисунке 2:

```
До преобразований: 255
После преобразований: 239
Program ended with exit code: 0
```

Рисунок 2 – Результат работы программы

1.4 Формулировка задачи 1.б

Реализовать по аналогии с предыдущим примером установку 7-го бита числа в 1.

1.5 Описание математической модели

В коде будут применяться битовые операции, такие как сдвиг влево, инверсия и побитовое ИЛИ. Переменные – инициализированы беззнаковым типом *unsigned* для работы с двоичными числами.

1.6 Реализация алгоритма на языке C++

Рассмотрим код:

```
int main() {
  setlocale(LC_ALL, "RUS");
  unsigned x = 191; // 8-разрядное двоичное число 10111111
  unsigned maska = 1; // 1=00000001 - 8-разрядная маяка
  cout << "До преобразований: " << x << endl;
  x = x | (maska << 6);
  cout << "После преобразований: " << x << endl;
}
```

Рисунок 3 – Реализация программы

Результат работы программы представлен на рисунке 4:

```
До преобразований: 191
После преобразований: 255
Program ended with exit code: 0
```

Рисунок 4 – Результат работы аналога предыдущего задания

1.7 Формулировка задачи 1.в

Реализовать код листинга, объяснить выводимый программой результат.

1.8 Описание математической модели

В коде будут применяться битовые операции, такие как сдвиг влево и вправо, побитовое И. Переменные – инициализированы беззнаковым типом unsigned для работы с двоичными числами.

1.9 Реализация алгоритма на языке C++

Рассмотрим код:

```cpp
int main(){

    unsigned int x = 25;
    const int n = sizeof(int) * 8; // = 32 - количество разрядов в числе типа int
    unsigned maska = (1 << (n - 1)); // 1 в старшем бите 32-разрядной сетки
    cout << "Исходная маска: " << bitset<n>(maska) << endl;
    cout << "Результат: ";
    for(int i = 1; i <= n; i++){
        cout << ((x & maska) >> (n - i));
        maska = maska >> 1; // смещение 1 в маске на разряд вправо
    }
    cout << endl << "Нажмите Enter для выхода...";
    cin.get();

    return 0;
}
```

Рисунок 5 – Реализация программы

Программа реализует алгоритм, который поочередно проверяет каждый бит 32-битного целого числа с помощью маски и выводит двоичное представление этого числа. Можно описать как процесс побитового сравнения числа с маской и сдвиг этой маски на каждом шаге для анализа следующего бита числа. Результатом программы будет двоичное представление исходного значения x (25) в обратном порядке.

```
Исходная маска: 10000000000000000000000000000000
Результат: 00000000000000000000000000011001
Нажмите Enter для выхода...
Program ended with exit code: 0
```

Рисунок 6 – Результат работы модернизированного кода для MacOS

ЗАДАНИЕ №2

2.1 Формулировка задачи 2.а

Написать код, реализованный в задании, с вводом произвольного набора до 8-ми чисел (со значениями от 0 до 7) и его быстрой сортировкой битовым массивом в виде числа типа unsigned char. Проверить работу программы.

2.2 Описание математической модели

Программа использует побитовое представление для хранения набора чисел от 0 до 7. В такой последовательности единичные биты показывают наличие в исходном наборе числа, равного номеру этого бита в последовательности. Последовательное считывание бит этой последовательности и вывод соответствующих индексов единичных битов позволит нам получить отсортированный набор данных.

2.3 Реализация алгоритма на языке C++

Рассмотрим код программы:

```
int main() {
    unsigned char mask = 0;// Битовое представление для сортировки
    int count;// Количество чисел для ввода (до 8)
    cout << "Введите количество чисел(1-8): ";
    cin >> count;
    if (count < 1 || count > 8) {
        cout << "Неверное количество чисел" << endl;
        return 1;
    }
    cout << "Введите числа(0-7): ";
    for (int i = 0; i < count; i++) {
        int num;
        cin >> num;
        if (num < 0 || num > 7) {
            cout << "Неверное число" << endl;
            return 1;
        }
        mask |= (1 << num);
    }
    cout << "Битовый массив: " << bitset<32>(mask) << endl;//Установление бита в битовый массиве

    cout << "Отсортированные числа: ";
    for (int i = 0; i < 8; i++) {
        if (mask & (1 << i)) {
            cout << i << " ";
        }
    }
    cout << endl;
    return 0;
}
```

Рисунок 7 – Реализация программы

Результат работы программы представлен на рисунке 8.

```
Введите количество чисел (от 1 до 8): 5
Введите числа (от 0 до 7): 3 7 1 0 4
Битовый массив: 00000000000000000000000010011011
Отсортированные числа: 0 1 3 4 7
Program ended with exit code: 0
```

Рисунок 8 – Результат работы алгоритма

2.4 Формулировка задачи 2.б

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

2.5 Описание математической модели

Программа работает по аналогии с предыдущей, однако для того, чтобы избежать ошибку при введении более 8 чисел или введении чисел, больших 7, будет использоваться тип беззнакового числа для битового массива с подходящим размером.

2.6 Реализация алгоритма на языке C++

Рассмотрим код программы:

```cpp
int main() {
    unsigned long long mask = 0;// Битовое представление для сортировки
    int count;// Количество чисел для ввода (до 64)
    cout << "Введите количество чисел(1-64): ";
    cin >> count;
    if (count < 1 || count > 64) {
        cout << "Неверное количество чисел" << endl;
        return 1;
    }
    cout << "Введите числа(0-63): ";
    for (int i = 0; i < count; i++) {
        int num;
        cin >> num;
        if (num < 0 || num > 63) {
            cout << "Неверное число" << endl;
            return 1;
        }
        mask |= (1ULL << num);// 1ULL - unsigned long long 1
    }
    cout << "Битовый массив: " << bitset<64>(mask) << endl;//Установление бита в битовый массиве

    cout << "Отсортированные числа: ";
    for (int i = 0; i < 64; i++) {
        if (mask & (1ULL << i)) {
            cout << i << " ";
        }
    }
    cout << endl;
    return 0;
}
```

Рисунок 9 – Реализация программы

Результат работы программы представлен на рисунке 10.

```
Введите количество чисел(1-64): 8
Введите числа(0-63): 7 2 1 5 9 11 8 13
Битовый массив: 0000000000000000000000000000000000000000000000000010101110100110
Отсортированные числа: 1 2 5 7 8 9 11 13
Program ended with exit code: 0
```

Рисунок 10 – Результат работы кода

2.7 Формулировка задачи 2.в

Исправить программу заданий 2.а-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.

2.8 Описание математической модели

Необходимо реализовать линейный массив таких чисел, который будет представлен одной непрерывной битовой последовательностью типа unsigned char, функцию добавления чисел в массив, функцию проверки вхождения числа в массив и функцию вывода данного массива в консоль. Массив mask будет использован как битовый массив, где каждый элемент массива представляет 8 чисел.

2.9 Реализация алгоритма на языке C++

Рассмотрим код программы:

```cpp
void addNum(unsigned char mask[], int num){
    mask[num / 8] |= (1 << (num % 8));
}

bool isNum(unsigned char mask[], int num){
    return(mask[num / 8] & (1 << (num % 8))) != 0;
}

void Print(unsigned char mask[]){
    for(int i = 0; i < 64; i++){
        if(isNum(mask, i)){
            cout << i << " ";
        }
    }
    cout << endl;
}

int main() {
    unsigned char mask[8] = {0};// Каждые 8 бит - 1 байт, 64 - чисел (бит)
    int count;// Количество чисел

    cout << "Введите количество чисел: ";
    cin >> count; // Количество вводимых чисел

    cout << "Введите числа: ";

    for (int i = 0; i < count; i++) {
        int num;
        cin >> num;
        addNum(mask, num);
    }

    Print(mask);// Вывод массива
    return 0;
}
```

Рисунок 11 – Реализация программы

Результат работы программы представлен на рисунке 11.

```
Введите количество чисел: 10
Введите числа: 1 45 32 41 56 2 19 23 2 10
1 2 10 19 23 32 41 45 56
Program ended with exit code: 0
```

Рисунок 12 – Результат работы программы

ЗАДАНИЕ №3

3.1 Формулировка задачи 3.а-3.б

Реализовать задачу сортировки числового файла с заданными условиями. Добавить в код возможность определения времени работы программы. Определить программно объем оперативной памяти, занимаемой битовым массивом.

3.2 Описание математической модели

Алгоритм сортировки большого объема данных представляет собой сортировку с использованием битового массива. Достаточно один раз считать содержимое файла, заполнив при этом битовый массив и на его основе быстро сформировать содержимое выходного файла в уже отсортированном виде. Для реализации алгоритма мы должны выделить массив mask размером, достаточным для хранения 10^7 чисел. Производится считывание данных из входного файла и устанавливаются биты в массиве mask в единицу, используя операции деления и умножения по модулю. После этого открывается выходной файл и проходит по массиву mask, выводя индексы битов, которые установлены в 1. В завершение измеряется время выполнения программы.

3.3 Реализация алгоритма на языке C++

Рассмотрим код программы:

```cpp
int main() {
    setlocale(LC_ALL, "RUS");
    clock_t start_time = clock();

    ifstream input_file("/Users/gwynbleidd/Desktop/Praktika 5.1 2/Praktika 5.1/input123.txt");//Открытие файл
        ввода
    if (!input_file) {
        cerr << "Ошибка открытия файла ввода" << endl;
        return 1;
    }

    vector<unsigned char> mask((MAX_BITS + BYTE_SIZE - 1) / BYTE_SIZE, 0);//Создание массива mask
    int num;
    int count = 0;
    while (input_file >> num) {
        if (num >= 0 && num < MAX_BITS) {
            mask[num / BYTE_SIZE] |= (1 << (num % BYTE_SIZE));//Установить соответствующий бит
            count++;
        }
    }
    input_file.close();

    ofstream output_file("/Users/gwynbleidd/Desktop/Praktika 5.1 2/Praktika 5.1/output123.txt");
    if (!output_file) {
        cerr << "Ошибка открытия файла вывода" << endl;
        return 1;
    }
    for (int i = 0; i < MAX_BITS; i++) { //Перебрать массив mask
        if (mask[i / BYTE_SIZE] & (1 << (i % BYTE_SIZE))) { //Проверить, установлен ли бит
            output_file << i << " "; //Вывести индекс установленного бита
        }
    }
    output_file.close();

    clock_t end_time = clock();
    double elapsed_time = static_cast<double>(end_time - start_time) / CLOCKS_PER_SEC;
    printf("Время работы: %.2f секунд\n", elapsed_time);
    return 0;
}
```

Рисунок 13 – Реализация программы

Часть данных файла ввода представлена на рисунке 14.

```
186 830 661 626 469 703 32 251
969
149
940
70
393
451
980
711
559
```

Рисунок 14 – Данных исходного файла

Часть отсортированных данных выходного файла представлена на рисунке 15.

```
1 4 15 17 22 27 28 32 34 35 38 41 43 44 45 47 49 51 52 53 55 59 61 62 63 66 70 74 77 79 82 83 85 94 95 98 100
    102 103 104 105 108 110 111 112 114 116 122 126 129 132 134 137 139 141 145 146 147 149 151 152 153 154
    155 157 163 165 167 169 170 180 181 182 183 185 186 188 189 198 206 209 217 218 219 220 221 222 226 229
    230 231 232 235 247 248 250 251 253 259 269 272 275 276 277 282 285 289 293 299 300 301 303 305 306 307
```

Рисунок 15– Данные выходного файла

Время работы программы 0, . Необходимо определить объем оперативной памяти, занимаемый битовым массивом. Код представлен на рисунке 16, результат подсчета на 17.

```
size_t memory_usage = mask.size() * sizeof(mask[0]);//Путем умножения кол-во элементов на размер каждого
cout << "Объем памяти: " << memory_usage << " бит" << endl;
return 0;
```

Рисунок 16 – Код, вычисляющий объем памяти

```
Время работы: 0.68 секунд
Объем памяти: 16777216 бит
Program ended with exit code: 0
```

Рисунок 17 – Результат выполнения программы

ВЫВОДЫ

В ходе выполнения практической работы были приобретены навыки работы с битовым представлением беззнаковых целых чисел и разработан эффективный алгоритм внешней сортировки с использованием битового массива. Полученные результаты свидетельствуют о том, что сортировка на основе битового массива значительно превосходит другие методы внешней сортировки по эффективности.

ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

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 08.09.2024).

3. Курс: Структуры и алгоритмы обработки данных. Часть 2 [Электронный ресурс]. https://online-edu.mirea.ru/course/view.php?id=4020 (дата обращения 04.09.2024)