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

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 5.1** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Работа с данными из файла»** | |
|  | |
| Выполнил студент группы ИКБО-10-23 | Харитонов А.Н. |
| Принял преподаватель | Макеева О.В. |

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

Москва 2024

# **Цель работы**

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

# **Ход работы**

**Задание 1**

Основная идея задания – познакомиться с битовыми операциями в языке C++.

Для реализации задания **1a** воспользуемся кодом из примера и протестируем его

#include <iostream>

int main(){

    unsigned char x = 255;

    unsigned char maska = 1;

    x = x & (~ (maska << 4));

    std::cout << (int) x;

}

При запуске программы видим 239, что удовлетворяет поставленной задаче.

Для реализации задания **1б** немного перепишем код задачи 1а и будем сдвигать биты маски до 7.

#include <iostream>

int main(){

    unsigned char x = 10; *//11111111*

    unsigned char maska = 1; *//00000001*

    x = x | (maska << 6); *// сдвиг до 7 бита*

    std::cout << (int) x;

}

В результате мы увидим 74, что также удовлетворяет условию задачи.

Для реализации задания **1в** перепишем код листинга.

int main(){

    SetConsoleCP(CP\_UTF8);

    SetConsoleOutputCP(CP\_UTF8);

    unsigned int x = 255;

    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; *// сдвигаем единицу*

    }

    cout << endl;

    system("pause");

    return 0;

}

Результатом будет 16 нулей и 8 единиц. Это достигается тем, что наша 1 в маске начинает сдвигаться в цикле и, пока она не дошла до 8 бита, ответ равен 0, ведь i-ый бит числа, при i > 8, равен 0.

**Задание 2**

Цель данного задания – познакомиться с сортировкой, используя битовые массивы.

**2а**: Реализуем программу, которая принимает на вход массив, длина которого не превышает 7, состоящий из уникальных символов, которые также не превышают 7.

Для решения поставленной задачи, объявим переменную bitmap типа unsigned char. Будем проходить по введенному массиву и ставить соответствующий бит в единицу.

int main(){

    int inArr[4] = {3, 7, 2, 0};

    unsigned char bitmap = 0;

    for (auto i : inArr){

        bitmap = bitmap | (1 << i); *//ставим соответствующий бит в 1*

    }

    for (int i = 0; i < 8; i++){

        if (bitmap & (1 << i)){

            std::cout << i << " ";

        }

    }

}

Результат будет “0 2 3 7”.

**2b**: Адаптируем код из примера 2a для типа unsigned long long.

Алгоритм схож, но теперь используем unsigned long long в качестве массива.

int main(){

    int inArr[12] = {9, 8, 7, 6, 5, 4, 0, 2, 10, 43, 14, 63};

    unsigned long long bitmap = 0;

    for (auto i : inArr){

        bitmap =  bitmap | (1ULL << i); *// устанавливаем бит в 1*

    }

    for (int i = 0; i < 64; i++){

        if ((long long)bitmap & (1ULL << i)){

            std::cout << i << " ";

        }

    }

}

Результатом будет “0 2 4 5 6 7 8 9 10 14 43 63”

**2в**: Адаптируем код 2б для массива чисел типа unsigned char.

Алгоритм также похож на предыдущий, но теперь для каждого значения у нас будет индекс в массиве и индекс в битовой маске.

int main(){

    int inArr[12] = {9, 8, 7, 6, 5, 4, 3, 2, 10, 43, 14, 17};

    unsigned char bitmap[8] = {0};

    for (auto i : inArr){

        bitmap[i / 8] = bitmap[i / 8] | (1 << i % 8);

    }

    for (int i = 0; i < 64; i++){

        if (bitmap[i / 8] & (1 << (i % 8))){

            std::cout << i << " ";

        }

    }

}

Результатом будет “2 3 4 5 6 7 8 9 10 14 17 43”

**Задание 3**

Упорядочить данную в файле последовательность чисел, не превышающих 10\*\*7, вывести затраченное время и оперативную память.

Для решения поставленной задачи был сгенерирован файл с 1000000 числами, которые не превышают 10\*\*7.

Алгоритм повторяет алгоритм для решения задания 2в, но теперь надо использовать массив длинной 10\*\*7 / 8 + 1, чтобы уместить все значения.

#include <iostream>

#include <fstream>

#include <vector>

#include <chrono>

#include <cmath>

#include <bitset>

#include <windows.h>

#include <psapi.h>

using namespace std;

void printMemoryUsage() {

    PROCESS\_MEMORY\_COUNTERS\_EX pmc;

    HANDLE hProcess = GetCurrentProcess();

    if (GetProcessMemoryInfo(hProcess, (PROCESS\_MEMORY\_COUNTERS\*)&pmc, sizeof(pmc))) {

        cout << "Memory usage: " << pmc.PrivateUsage / (1024 \* 1024) << " MB" << endl;

    } else {

        cerr << "Failed to get memory usage." << endl;

    }

}

void makeTask(){

    const int bytes = 8;

    int arraySize = pow(10, 7) / bytes + 1;

    vector<unsigned char> bitmap(arraySize, 0); *// битовый массив*

    ifstream inputFile("input.txt");

    int number;

    while (inputFile >> number){

        bitmap[number / bytes] = bitmap[number / bytes] | (1ULL << number % bytes); *// установка заданного бита в 1*

    }

    inputFile.close();

    ofstream outputFile("output.txt");

    for (int i = 0; i < arraySize \* bytes; i++){

        if (bitmap[i / bytes] & (1ULL << i % bytes)){

            outputFile << i << endl;

        }

    }

}

int main(){

    auto start = chrono::high\_resolution\_clock::now(); *//время начала*

    makeTask();

    printMemoryUsage(); *//вывод занятой оперативной памяти*

    auto end = chrono::high\_resolution\_clock::now(); *// время конца*

    chrono::duration<double> timeGone = end - start;

    cout << "TIME - " << timeGone.count() << endl;

}

В результате будет показано затраченное время, затраченная оперативная память, которая не превышает 1МБ, и файл с отсортированными значениями.

# **Вывод**

В ходе работы я познакомился с побитовыми операциями в языке С++, а также написал эффективный алгоритм внешней сортировки на основе битового массива.