Міністерство освіти і науки України
Західноукраїнський національний університет
Факультет комп’ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

Звіт
про виконання лабораторної роботи № 9
з дисципліни «Високопродуктивні обчислення»

Виконав:

Студент групи КН-43

Гнатів С. В.

Тернопіль - 2024

Лабораторна робота № 9

Тема: Сортування з використанням OpenMP.

Мета:Вивчення основних алгоритмів сортування. Використання директив розпаралелення.

Завдання

1. Реалізувати алгоритм сортування «бульбашкою» у його послідовній версії для однопроцесорних систем, паралельної версії для багатопроцесорних систем, використовуючи #pragma omp parallel for OpenMP і модифікацію методу сортування «бульбашкою», відомий в літературі як метод парноїнепарної перестановки (the odd-even transposition method).Сутність модифікації полягає в тому, що в алгоритм сортування вводять два різні правила виконання ітерацій методу: залежно від парності або непарності номера ітерації сортування для обробки вибираються елементи з парними або непарними індексами відповідно, порівняння виділених значень завжди здійснюється з їх правими сусідніми елементами. Отже, на всіх непарних ітераціях порівнюються пари: (a1, a2), (a3, a4), ..., (an-1,an) (при парному n), а на парних ітераціях обробляються елементи: (a2, a3), (a4, a5), ..., (an-2,an-1). Після n-кратного повторення ітерацій сортування вихідний набір даних виявляється впорядкованим.
2. Обчислити час роботи кожної із програм у всіх випадках, протестувати і порівняти прискорення паралельного та послідовного алгоритму.

Хід роботи

Реалізація алгоритму сортування «бульбашкою» у його послідовній версії продемонстровано на рисунку 1.1.

```cpp
void bubbleSort(std::vector<int>& arr) {
    int n = arr.size();
    std::cout << "Sequential Bubble Sort process:" << std::endl;
    for (int i = 0; i < n - 1; ++i) {
        for (int j = 0; j < n - i - 1; ++j) {
            if (arr[j] > arr[j + 1]) {
                std::cout << "Swapping " << arr[j] << " and " << arr[j + 1] << ": \n";
                std::swap(arr[j], arr[j + 1]);
                std::cout <<"Array element swap:\t";
                printArray(arr);
            }
        }

        if (i < n - 2) {
            std::cout << "End of iteration " << i + 1 << ": ";
            printArray(arr);
        }
    }
}
```

Рисунок 1.1 - Алгоритм сортування «бульбашкою» у його послідовній версії.

Реалізація алгоритму сортування «бульбашкою» у паралельній версії продемонстровано на рисунку 1.2.

```cpp
void parallelBubbleSort(std::vector<int>& arr) {
    int n = arr.size();
    bool swapped;

    std::cout << "Parallel Bubble Sort process:" << std::endl;
    for (int i = 0; i < n - 1; ++i) {
        swapped = false;

        #pragma omp parallel for
        for (int j = 0; j < n - i - 1; ++j) {
            if (arr[j] > arr[j + 1]) {
                #pragma omp critical
                {
                    std::cout << "Swapping elements at indices " << j << " and " << j + 1
                              << " (values " << arr[j] << " and " << arr[j + 1] << ")" << std::endl;
                }
                std::swap(arr[j], arr[j + 1]);
                swapped = true;
            }
        }
        std::cout << "Array after iteration " << i + 1 << ": ";
        printArray(arr);

        if (!swapped) break;
    }
}
```

Рисунок 1.2 - Алгоритм сортування «бульбашкою» у паралельній версії.

Реалізація алгоритму сортування «бульбашкою» у його послідовній версії продемонстровано на рисунку 1.3.

```cpp
void oddEvenSort(std::vector<int>& arr) {
    int n = arr.size();
    int phase, i, temp;
    std::cout << "Odd-Even Transposition Sort process:" << std::endl;
    for (phase = 0; phase < n; phase++) {
        if (phase % 2 == 0) {
            std::cout << "Even phase: \n";
            for (i = 1; i < n; i += 2) {
                if (arr[i - 1] > arr[i]) {
                    std::cout << "Swapping " << arr[i - 1] << " and " << arr[i] << " at indices "
                              << i - 1 << " and " << i << "\n";
                    temp = arr[i - 1];
                    arr[i - 1] = arr[i];
                    arr[i] = temp;
                }
            }
        } else {
            std::cout << "Odd phase: \n";
            for (i = 1; i < n - 1; i += 2) {
                if (arr[i] > arr[i + 1]) {
                    std::cout << "Swapping " << arr[i] << " and " << arr[i + 1] << " at indices "
                              << i << " and " << i + 1 << "\n";
                    temp = arr[i];
                    arr[i] = arr[i + 1];
                    arr[i + 1] = temp;
                }
            }
        }
        std::cout << "Result phase sorting: ";
        printArray(arr);
    }
}
```

Рисунок 1.3 - Метод парноїнепарної перестановки.

Реалізація алгоритму сортування «бульбашкою» у паралельній версії продемонстровано на рисунку 1.4.

```cpp
int main() {
    std::vector<int> arr = {5, 3, 8, 4, 2, 10, 2,2323, 32, 3 };
    std::vector<int> arrCopy;
    std::cout << "Unsorted array: ";
    printArray(arr); //std::cout << "--------------------------------------------------------------------------------"<<std
    arrCopy = arr;
    auto start = std::chrono::high_resolution_clock::now();
    bubbleSort(arrCopy);
    auto end = std::chrono::high_resolution_clock::now();
    std::chrono::duration<double> duration = end - start;
    std::cout << "Bubble Sort sorted array: ";
    printArray(arrCopy);
    std::cout << "Bubble Sort execution time: " << duration.count() << " seconds\n\n"; //std::cout << "--------------
    arrCopy = arr;
    start = std::chrono::high_resolution_clock::now();
    parallelBubbleSort(arrCopy);
    end = std::chrono::high_resolution_clock::now();
    duration = end - start;
    std::cout << "Parallel Bubble Sort sorted array: ";
    printArray(arrCopy);
    std::cout << "Parallel Bubble Sort execution time: " << duration.count() << " seconds\n\n";//std::cout << "-------
    arrCopy = arr;
    start = std::chrono::high_resolution_clock::now();
    oddEvenSort(arrCopy);
    end = std::chrono::high_resolution_clock::now();
    duration = end - start;
    std::cout << "Odd-even transposition sorted array: ";
    printArray(arrCopy);
    std::cout << "Execution time: " << duration.count() << " seconds\n";
    return 0;
}
```

Рисунок 1.3 - Метод main.

Результат виконання для кожного для кожного методу продемонстровано на рисунках 1.4, 1.5, 1.6,

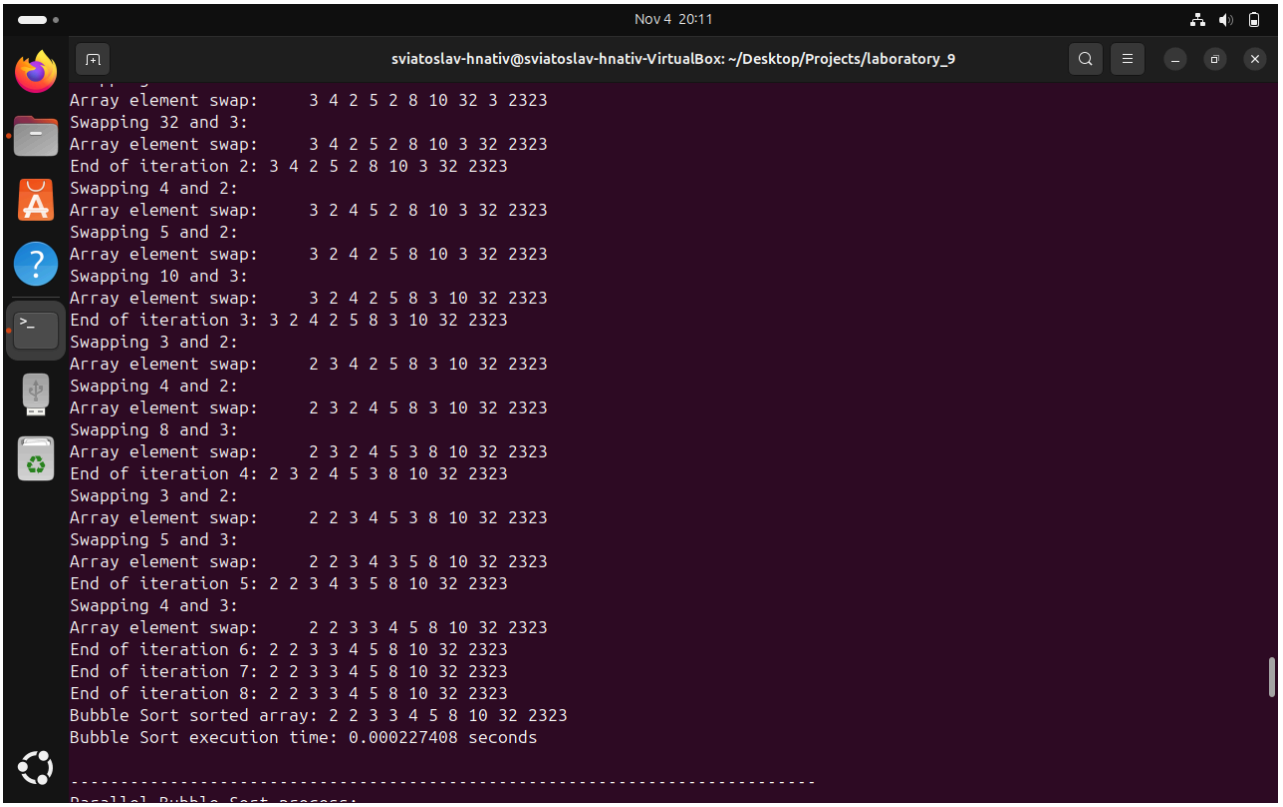

Рисунок 1.4 - Результат сортування «бульбашкою» у його послідовній версії.

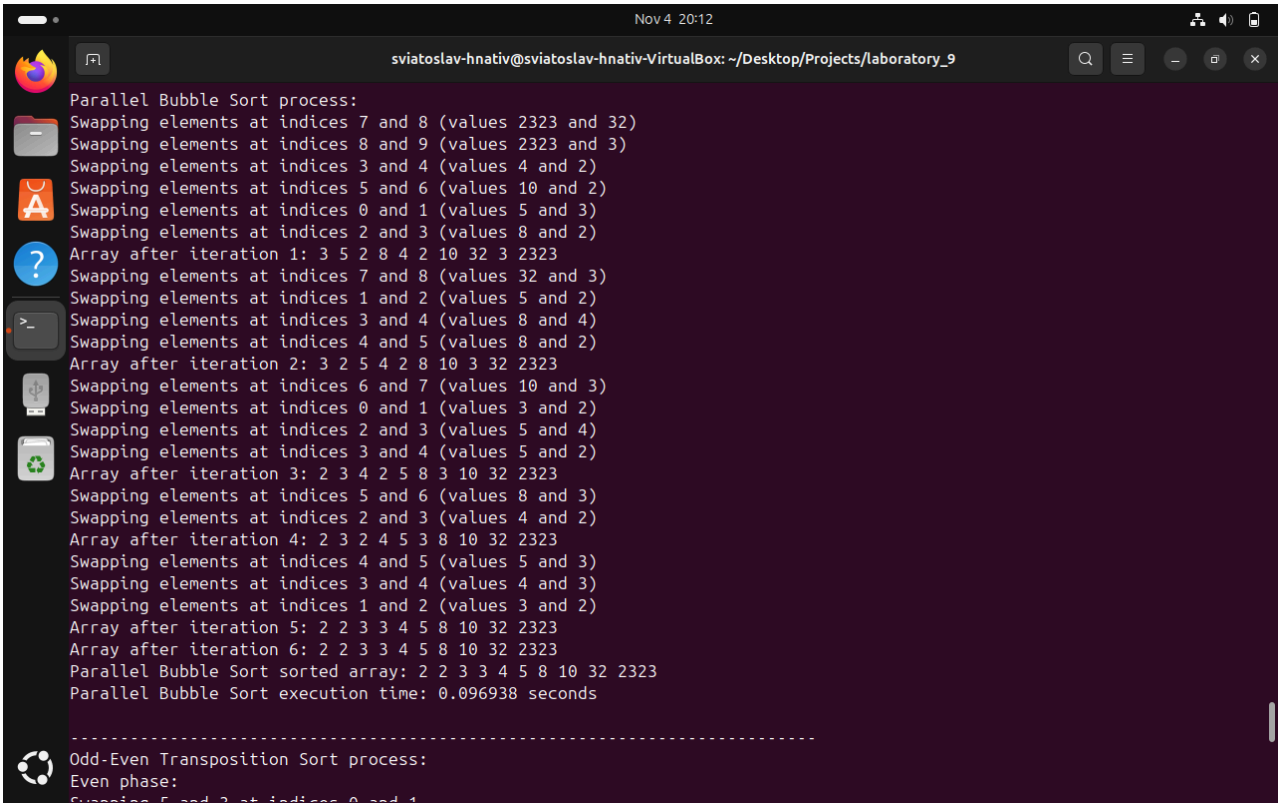

Рисунок 1.5 -Результат сортування «бульбашкою» у паралельній версії.

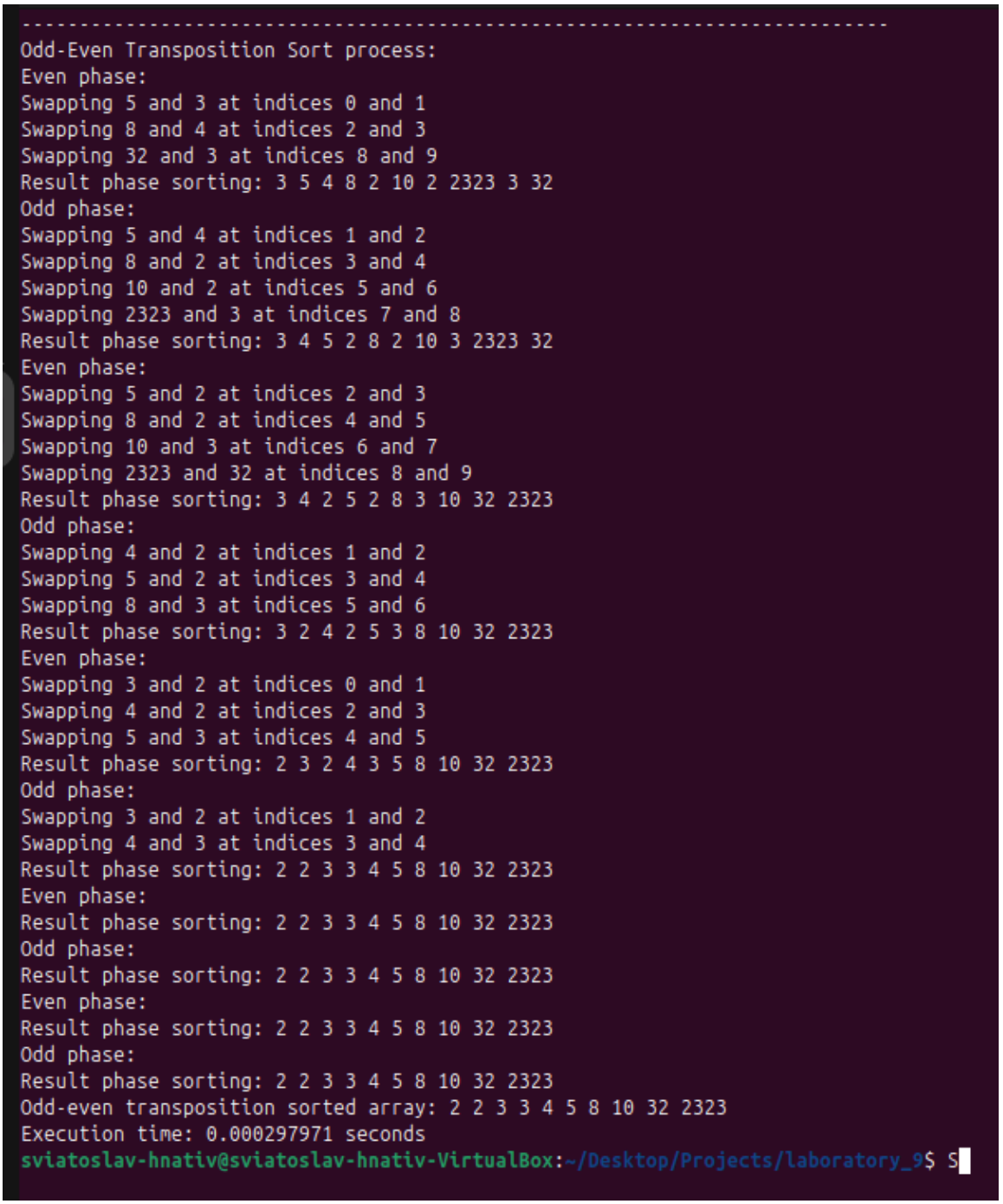

Рисунок 1.6 - Результат методу парноїнепарної перестановки.

Висновки:

Під час виконання лабораторної роботи було отримано навички роботи бібліотекою OpenMP та реалізовано методи сортування за допомогою даної директиви