**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

**Кафедра безопасности информационных систем**

**ОТЧЁТ**

по практической работе № 4 на тему:   
**«Сортировка числовых массивов. Некоторые**

**методы сортировки»**

по дисциплине «Алгоритмы и структуры данных»

Выполнил: студент группы ИСТ-312, Серафимович Г. П.

«10» октября 2024 г. \_\_\_\_\_\_\_\_\_\_\_/ Серафимович Г. П./

Принял: к.ф.-м.н., доцент, И.А. Моисеев

«10» октября 2024 г. \_\_\_\_\_\_\_\_\_\_\_/ И.А. Моисеев /

**Основная часть**

**Цель работы:**

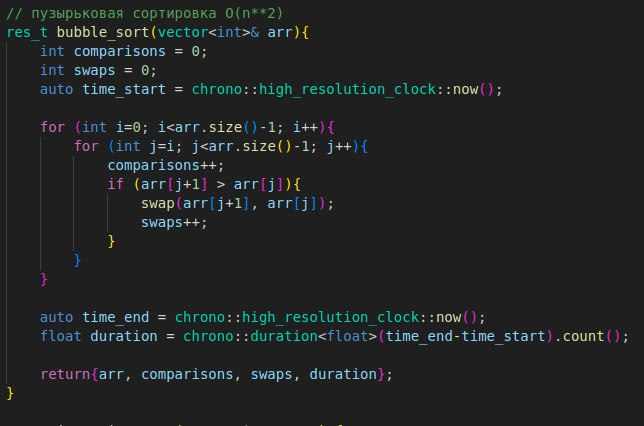
Целью данной работы является реализация и сравнение трех алгоритмов сортировки: пузырьковой сортировки, сортировки вставками и сортировки выбором. Программа должна считывать массив чисел из файла, сортировать его с использованием указанных алгоритмов и выводить отсортированные массивы на экран.

**Результаты выполнения работы:**

Были реализованы следующие методы:

1. Пузырьковая сортировка
2. Сортировка вставками
3. Сортировка выбором

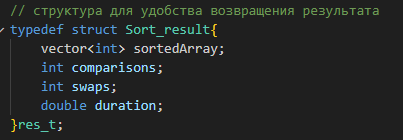
Был реализован метод пузырьковой сортировки (Рисунок 1)



**Рисунок 1. Функция пузырьковой сортировки**

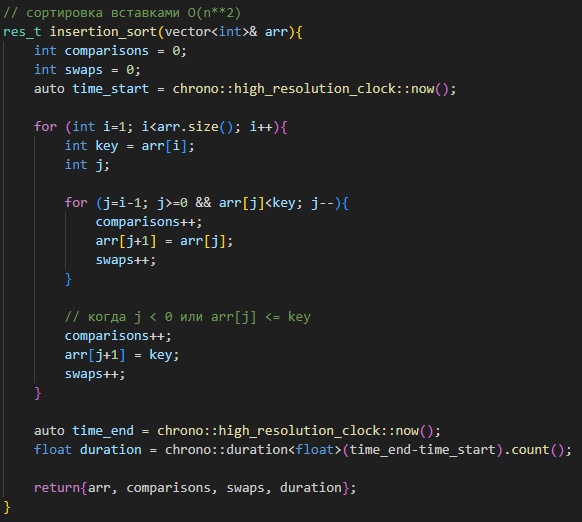
В этой функции все просто, цикл и вложенный цикл в которых два элемента сравниваются и меняются местами при необходимости, таким образом с помощью вложенного цикла for определяется минимальное число, оно перемещается в конец, вложенный цикл запускается ровно столько сколько чисел в массиве, таким образом все грамотно работает.

Самое важное в возвращении функции, она возвращает несколько значений, которые удовлетворяют структуре Sort\_result (Рисунок 2), это нужно для удобного счета сравнений, кол-ва обмена значений между переменными(swap) и времени выполнения сортировки.



**Рисунок 2. структура Sort\_result**

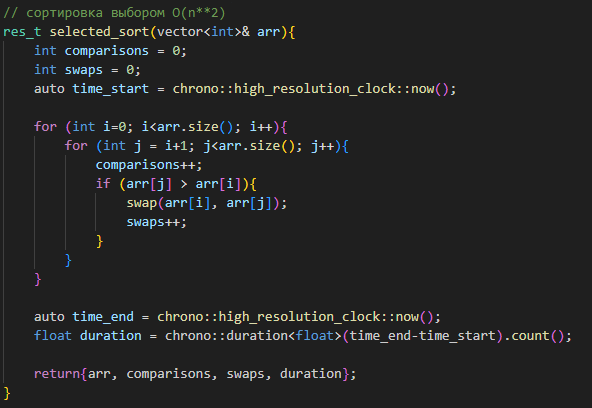
Был реализован метод сортировки вставками (Рисунок 3)



**Рисунок 3. Функция сортировки вставками**

Основной цикл проходит по всем элементам массива, начиная со второго, и для каждого элемента (ключа) происходит сравнение с предыдущими элементами, чтобы вставить его в правильную позицию. Если элемент меньше текущего, все последующие элементы сдвигаются вправо, увеличивая счетчики сравнений и обменов. В конце функции возвращается отсортированный массив вместе с количеством сравнений, обменов и временем выполнения.

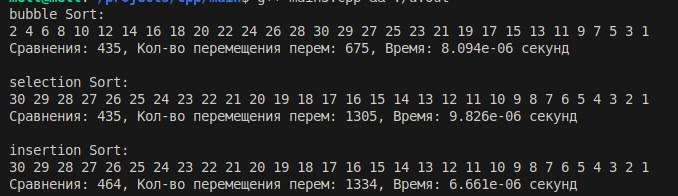
Был реализован метод сортировки выбором (Рисунок 4)



**Рисунок 4. Функция сортировки выбором**

Функция реализует сортировку выбором для массива целых чисе. Она проходит по каждому элементу массива и сравнивает его с последующими элементами, увеличивая счетчик сравнений. Если найден элемент, который больше текущего, происходит обмен, и увеличивается счетчик обменов. Время выполнения функции измеряется с помощью chrono, и в конце возвращается отсортированный массив вместе с количеством сравнений, обменов и временем выполнения.

Общий вывод для всех методов сортировки (Рисунок 5)



**Рисунок 5. Вывод сортировок**

**Выводы:**

Я многое понял из реализации данной практической работы, все три алгоритма сортировки успешно реализованы и работают корректно, сортируя массив чисел. Несмотря на то, что все три алгоритма имеют одинаковую временную сложность O(n²), их производительность может различаться в зависимости от структуры входных данных. **Приложение**

Листинг программы:

#include <iostream>

#include <fstream>

#include <vector>

#include <chrono>

using namespace std;

// cтруктура для удобства возвращения результата

typedef struct Sort\_result{

vector<int> sortedArray;

int comparisons;

int swaps;

double duration;

}res\_t;

// пузырьковая сортировка O(n\*\*2)

res\_t bubble\_sort(vector<int>& arr){

int comparisons = 0;

int swaps = 0;

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

for (int i=0; i<arr.size()-1; i++){

for (int j=i; j<arr.size()-1; j++){

comparisons++;

if (arr[j+1] > arr[j]){

swap(arr[j+1], arr[j]);

swaps++;

}

}

}

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

float duration = chrono::duration<float>(time\_end-time\_start).count();

return{arr, comparisons, swaps, duration};

}

res\_t insertion\_sort(vector<int>& arr) {

int comparisons = 0;

int swaps = 0;

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

for (int i = 1; i < arr.size(); i++) {

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] < key) {

comparisons++;

arr[j + 1] = arr[j];

j--;

swaps++;

}

comparisons++;

arr[j + 1] = key;

swaps++;

}

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

float duration = chrono::duration<float>(time\_end - time\_start).count();

return {arr, comparisons, swaps, duration};

}

// сортировка выбором O(n\*\*2)

res\_t selected\_sort(vector<int>& arr){

int comparisons = 0;

int swaps = 0;

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

for (int i=0; i<arr.size(); i++){

for (int j = i+1; j<arr.size(); j++){

comparisons++;

if (arr[j] > arr[i]){

swap(arr[i], arr[j]);

swaps++;

}

}

}

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

float duration = chrono::duration<float>(time\_end-time\_start).count();

return{arr, comparisons, swaps, duration};

}

void print\_result(const string& name, const res\_t& result){

cout << name << ":\n";

for (int num : result.sortedArray){ cout << num << " "; }

cout << "\nСравнения: " << result.comparisons << ", Кол-во перемещения перем: " << result.swaps

<< ", Время: " << result.duration << " секунд\n\n";

}

int main(){

// открываю файл

ifstream file("input.txt");

// проверка

if (!file){ cout << "Ошибка открытия файла" << endl; return 1; }

vector<int> arr; // массив

int number;

// читаем числа

while (file >> number){ arr.push\_back(number); }

file.close();

vector<int> original\_arr = arr; // копия массива

// сортировки

print\_result("bubble Sort", bubble\_sort(original\_arr));

original\_arr = arr;

print\_result("selection Sort", selected\_sort(original\_arr));

original\_arr = arr;

print\_result("insertion Sort", insertion\_sort(original\_arr));

return 0;

}