Assignment – 9

1. Median

#include <iostream>

using namespace std;

// definisi fungsi main

int main() {

    // deklarasi variabel

    int n, temp, min;

    float med = 0;

    // input jumlah nilai (N)

    cout << "Jumlah nilai: "; cin >> n;

    // deklarasi array

    float arr[n];

    // input nilai

    cout << "Nilai: ";

    for (int i = 0; i < n; i++)

        cin >> arr[i];

    // mengurutkan data dengan metode selection sort

    for (int i = 0; i < n; i++) {

        min = i;

        for (int j = i + 1; j < n; j++) {

            if(arr[j] < arr[min]) {

                min = j;

            }

        }

        temp = arr[i];

        arr[i] = arr[min];

        arr[min] = temp;

    }

    // print data yang sudah diurutkan

    cout << "Data Terurut: ";

    for (int i = 0; i < n; i++)

        cout << arr[i] << " ";

    // mencari median berdasarkan beberapa kondisi

    if(n == 1) med = arr[0];    // nilai hanya 1

    else if (n % 2 == 0) med = (arr[(n / 2) - 1] + arr[n / 2]) / 2;     // jumlah nilai genap

    else med = arr[n / 2];      // jumlah nilai ganjil

    // print median

    cout << "\nMedian: " << med;

    return 0;

}

2. Sorting Methods Running Time

// library bits/stdc++.h menyediakan fungsi-fungsi standar yang telah disediakan oleh C++ termasuk iostream dan chrono

// chrono menyediakan fungsi-fungsi yang berhubungan dengan waktu dan pada program ini digunakan untuk menghitung running time dari beberapa metode sorting

#include <bits/stdc++.h>

// using namespace std tidak digunakan karena program ini menggunakan cukup banyak standard library sehingga menyebabkan munculnya ambigu pada fungsi-fungsi yang menggunakan awalan std::

int main() {

    // deklarasi variabel

    int n, i, temp, min;

    // input jumlah bilangan acak yang ingin diurutkan

    std::cout << "\nInsert number of random data: "; std::cin >> n;

    // jika input adalah bilangan negatif

    if(n < 0) std::cout << "Invalid number, please enter positive integers" << std::endl;

    // deklarasi array untuk menampung bilangan acak 1 sampai n

    int arr[n];

    // deklarasi array untuk menampung nilai running time dari tiap metode sorting

    double time[4];

    // membuat bilangan acak sebanyak n bilangan menggunakan fungsi rand() yang akan mengambil sebarang angka secara acak dan belum terurut

    for (int i = 0; i < n; i++)

        arr[i] = rand();

    //-----------------------------------------------------------------------------//

    // fungsi dari library chrono untuk mengawali perhitungan waktu

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

    // insertion sort (ascending)

    for(int j = 1; j < n; j++) {

        i = j - 1;

        temp = arr[j];

        while(arr[i] > temp && i >= 0) {

            arr[i + 1] = arr[i];

            i--;

        }

        arr[i + 1] = temp;

    }

    // akhir perhitungan waktu

    auto finish = std::chrono::*high\_resolution\_clock*::now();

    // menghitung waktu yang dibutuhkan dengan waktu akhir dikurangi waktu awal

    std::chrono::duration<long double> elapsed = finish - start;

    // memasukkan nilai waktu ke array time sesuai indeks

    time[0] = elapsed.count();

    //-----------------------------------------------------------------------------//

    // memasukkan bilangan acak baru ke array

    for (int i = 0; i < n; i++)

        arr[i] = rand();

    // awal perhitungan waktu baru

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

    // insertion sort (descending)

    for(int j = 1; j < n; j++) {

        i = j - 1;

        temp = arr[j];

        while(arr[i] < temp && i >= 0) {

            arr[i + 1] = arr[i];

            i--;

        }

        arr[i + 1] = temp;

    }

    // akhir perhitungan waktu

    finish = std::chrono::*high\_resolution\_clock*::now();

    // hitung waktu yang dibutuhkan

    elapsed = finish - start;

    // input waktu yang dibutuhkan sesuai indeks

    time[1] = elapsed.count();

    //-----------------------------------------------------------------------------//

    // memasukkan bilangan acak baru ke array

    for (int i = 0; i < n; i++)

        arr[i] = rand();

    // awal perhitungan waktu baru

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

    // selection sort (ascending)

    for (int i = 0; i < n; i++) {

        min = i;

        for (int j = i + 1; j < n; j++) {

            if(arr[j] < arr[min]) {

                min = j;

            }

        }

        temp = arr[i];

        arr[i] = arr[min];

        arr[min] = temp;

    }

    // akhir perhitungan waktu

    finish = std::chrono::*high\_resolution\_clock*::now();

    // hitung waktu yang dibutuhkan

    elapsed = finish - start;

    // input waktu yang dibutuhkan sesuai indeks

    time[2] = elapsed.count();

    //-----------------------------------------------------------------------------//

    // memasukkan bilangan acak baru ke array

    for (int i = 0; i < n; i++)

        arr[i] = rand();

    // awal perhitungan waktu baru

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

    // selection sort (descending)

    for (int i = 0; i < n; i++) {

        min = i;

        for (int j = i + 1; j < n; j++) {

            if(arr[j] > arr[min]) {

                min = j;

            }

        }

        temp = arr[i];

        arr[i] = arr[min];

        arr[min] = temp;

    }

    // akhir perhitungan waktu

    finish = std::chrono::*high\_resolution\_clock*::now();

    // hitung waktu yang dibutuhkan

    elapsed = finish - start;

    // input waktu yang dibutuhkan sesuai indeks

    time[3] = elapsed.count();

    //-----------------------------------------------------------------------------//

    // array of string untuk menampung nama jenis sorting yang akan di-print ke tabel

    std::*string* sort[4] = {"Insertion Sort (Ascending)",

                    "Insertion Sort (Descending)",

                    "Selection Sort (Ascending)",

                    "Selection Sort (Descending)"};

    // print tabel

    std::cout << "\nNo\tSorting Method\t\t\tRunning Time\n";

    for (int i = 0; i <= 3; i++)

        std::cout << i + 1 << ")  "<< sort[i] << "\t\t" << time[i] << " s" << std::endl;

    // print spasi

    std::cout << std::endl;

    return 0;

}

3. STL C++ sort() Running Time

#include <bits/stdc++.h>

// salah satu fungsi standard template library (STL) yang disediakan library bits/stdc++.h digunakan pada program ini, yaitu fungsi sort() untuk mengurutkan data array

// fungsi chrono juga digunakan pada program ini untuk menghitung running time

using namespace std;

// definisi fungsi main

int main() {

    // deklarasi variabel

    int n, temp, i;

    float med = 0;

    // input jumlah nilai (N)

    cout << "Insert number of random data: "; cin >> n;

    // deklarasi array

    float arr[n];

    //-----------------------------------------------------------------------------//

    // membuat bilangan acak sebanyak n bilangan menggunakan fungsi rand() yang akan mengambil sebarang angka secara acak dan belum terurut

    for (int i = 0; i < n; i++)

        // (% n + 1) digunakan agar nilai yang diperoleh berkisar antara 1 sampai n

        arr[i] = rand() % n + 1;

    // fungsi dari library chrono untuk mengawali perhitungan waktu

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

    // mengurutkan data dengan metode ascending insertion sort

    for(int j = 1; j < n; j++) {

        i = j - 1;

        temp = arr[j];

        while(arr[i] > temp && i >= 0) {

            arr[i + 1] = arr[i];

            i--;

        }

        arr[i + 1] = temp;

    }

    // mencari median berdasarkan beberapa kondisi

    if(n == 1) med = arr[0];    // nilai hanya 1

    else if (n % 2 == 0) med = (arr[(n / 2) - 1] + arr[n / 2]) / 2;     // jumlah nilai genap

    else med = arr[n / 2];      // jumlah nilai ganjil

    // akhir perhitungan waktu

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

    // menghitung waktu yang dibutuhkan dengan waktu akhir dikurangi waktu awal

    chrono::duration<long double> elapsed = finish - start;

    // print hasil perhitungan

    cout << "Insertion Sort\t-> " << elapsed.count() << " s\t(Median: " << med << ")\n";

    //-----------------------------------------------------------------------------//

    // memasukkan bilangan acak baru ke array

    for (int i = 0; i < n; i++)

        arr[i] = rand() % n + 1;

    // awal perhitungan waktu baru

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

    // mengurutkan data dengan fungsi

    sort(arr, arr+n);

    // mencari median berdasarkan beberapa kondisi

    if(n == 1) med = arr[0];    // nilai hanya 1

    else if (n % 2 == 0) med = (arr[(n / 2) - 1] + arr[n / 2]) / 2;     // jumlah nilai genap

    else med = arr[n / 2];      // jumlah nilai ganjil

    // akhir perhitungan waktu

    finish = chrono::*high\_resolution\_clock*::now();

    // hitung waktu yang dibutuhkan

    elapsed = finish - start;

    // print hasil perhitungan

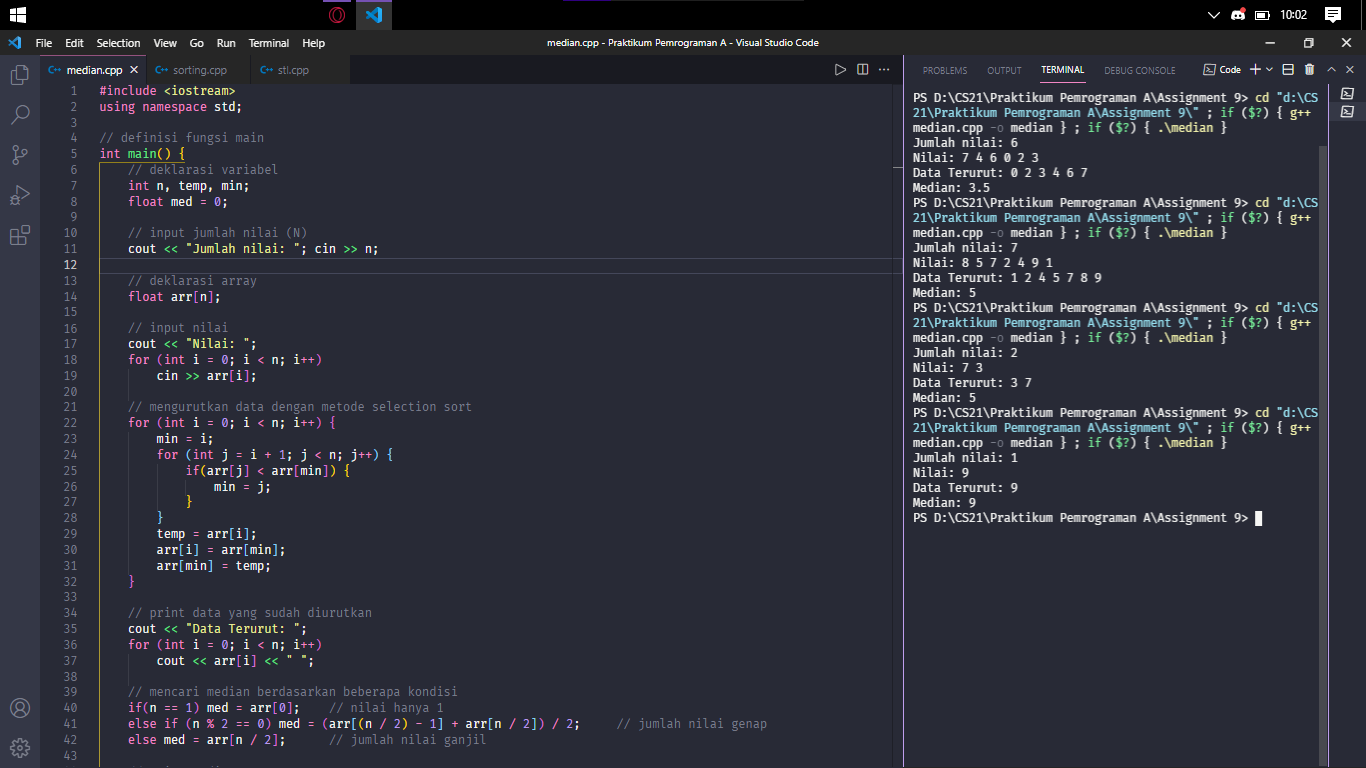
    cout << "STL Sort\t-> " << elapsed.count() << " s\t(Median: " << med << ")\n";

    return 0;

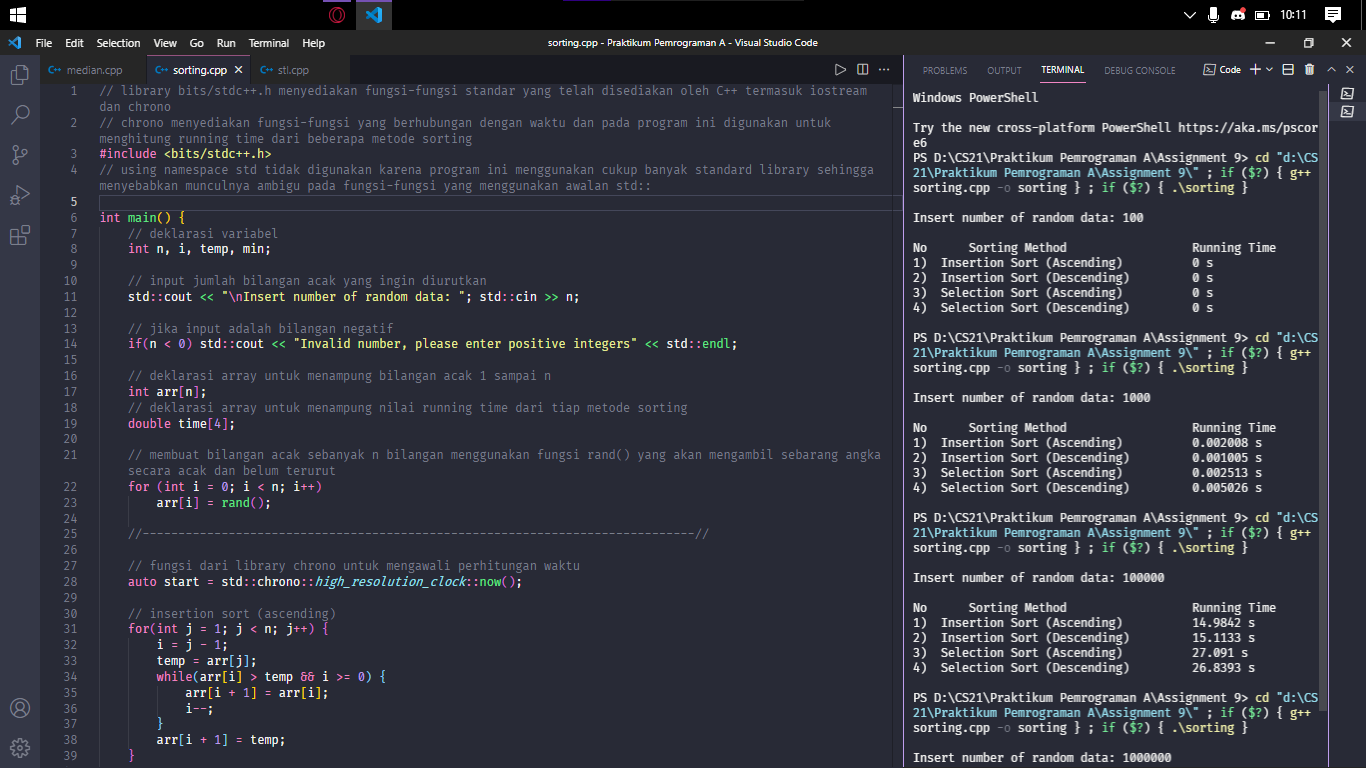
}

Screenshot

1. Median



2. Sorting Methods Running Time



3. STL C++ sort() Running Time

