**LAPORAN PRATIKUM STRUKTUR DATA**

**JOBSHEET 10**



**Disusun Oleh :**

**NAJWA ELDIARA OWILIA TIKSA**

**BP/NIM : 2023/23343079**

**Dosen Pengampu :**

**Randi Proska Sandra, S.Pd, M.Sc**

**Kode Kelas : 202323430157**

**PROGRAM STUDI S1 INFORMATIKA**

**DEPARTEMEN ELEKTRONIKA**

**FAKULITAS TEKNIK**

**UNIVERSITAS NEGERI PADANG**

**2023**

**Shell sort**

1. **Source Code**

//Created by 23343079\_Najwa Eldiara Owilia Tiksa

#include <stdio.h>

// Fungsi untuk melakukan pengurutan dengan Shell Sort

void shellSort(int arr[], int n) {

// Mulai dengan jarak yang besar, kemudian kurangi jaraknya

for (int jarak = n / 2; jarak > 0; jarak /= 2) {

// Lakukan pengurutan insertion sort untuk elemen-elemen dengan jarak yang diberikan

for (int i = jarak; i < n; i++) {

int elemen = arr[i];

int j;

for (j = i; j >= jarak && arr[j - jarak] > elemen; j -= jarak) {

arr[j] = arr[j - jarak];

}

arr[j] = elemen;

}

}

}

// Fungsi untuk mencetak array

void cetakArray(int arr[], int ukuran) {

for (int i = 0; i < ukuran; i++) {

printf("%d ", arr[i]);

}

printf("\n");

}

int main() {

int arr[] = {30, 24, 73, 69, 11};

int ukuran = sizeof(arr) / sizeof(arr[0]);

printf("Array sebelum diurutkan: \n");

cetakArray(arr, ukuran);

shellSort(arr, ukuran);

printf("Array setelah diurutkan: \n");

cetakArray(arr, ukuran);

return 0;

}

1. **Penjelasan Program**

Algoritma Shell Sort

Shell Sort adalah algoritma pengurutan yang memperbaiki kekurangan dari Insertion Sort. Insertion Sort bekerja dengan baik untuk data yang hampir terurut tetapi kurang efisien untuk data acak besar. Shell Sort mengatasi hal ini dengan:

Membandingkan elemen-elemen yang dipisahkan oleh jarak yang besar dan menguranginya secara bertahap.

Pada setiap iterasi, jarak dikurangi hingga jarak menjadi 1, yang kemudian beralih menjadi Insertion Sort tetapi pada array yang sebagian besar sudah diurutkan.

Proses pengurangan jarak ini membantu memindahkan elemen-elemen ke posisi yang lebih mendekati posisi akhir mereka lebih cepat dibandingkan hanya menggunakan Insertion Sort pada seluruh array dari awal.

Contoh Langkah Shell Sort

Misalkan kita memiliki array [30, 24, 73, 69, 11]:

Langkah pertama dengan jarak = 2:

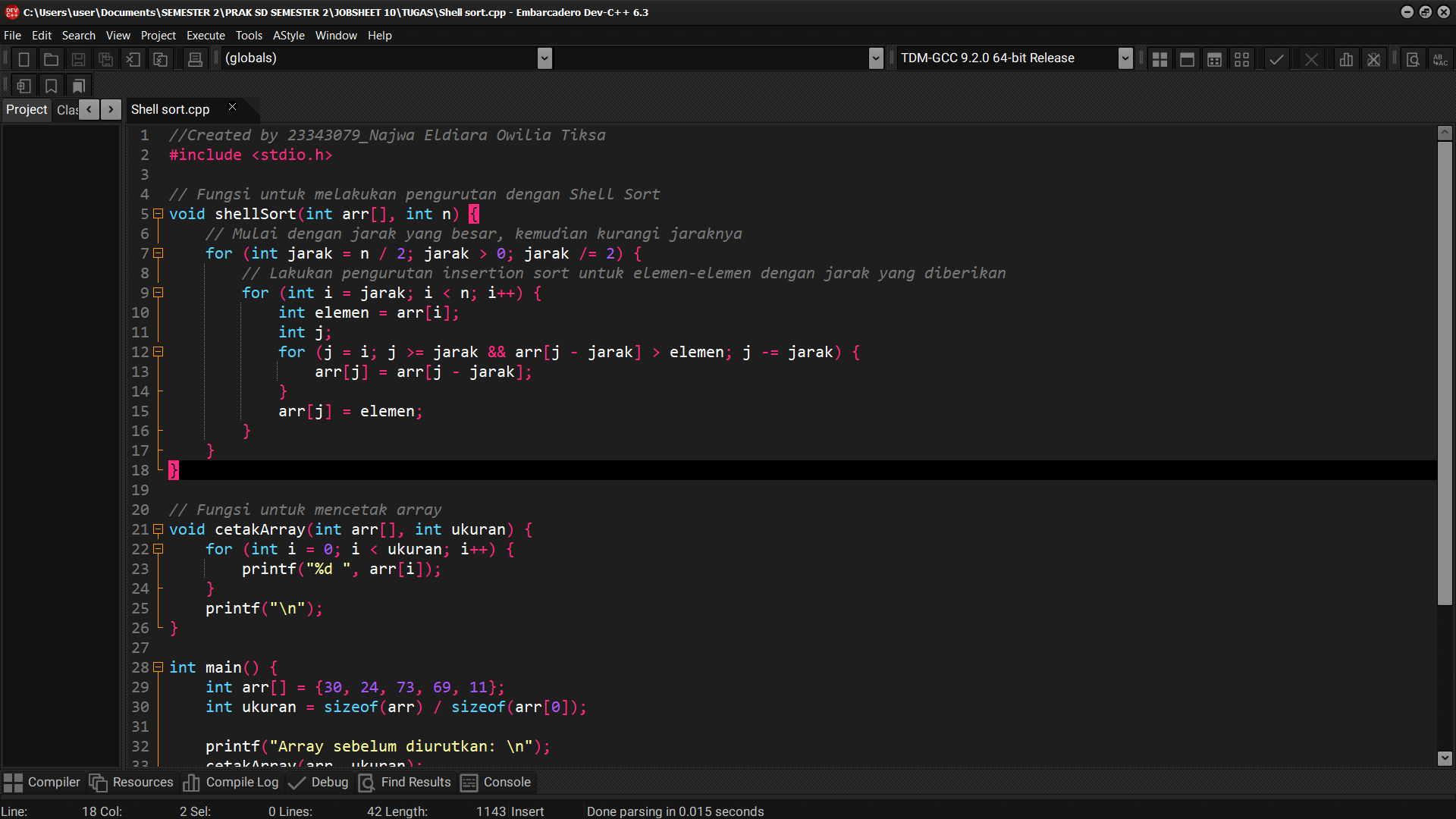
Membandingkan dan menukar elemen dengan jarak 2: [30, 24, 73, 69, 11] menjadi [30, 24, 11, 69, 73].

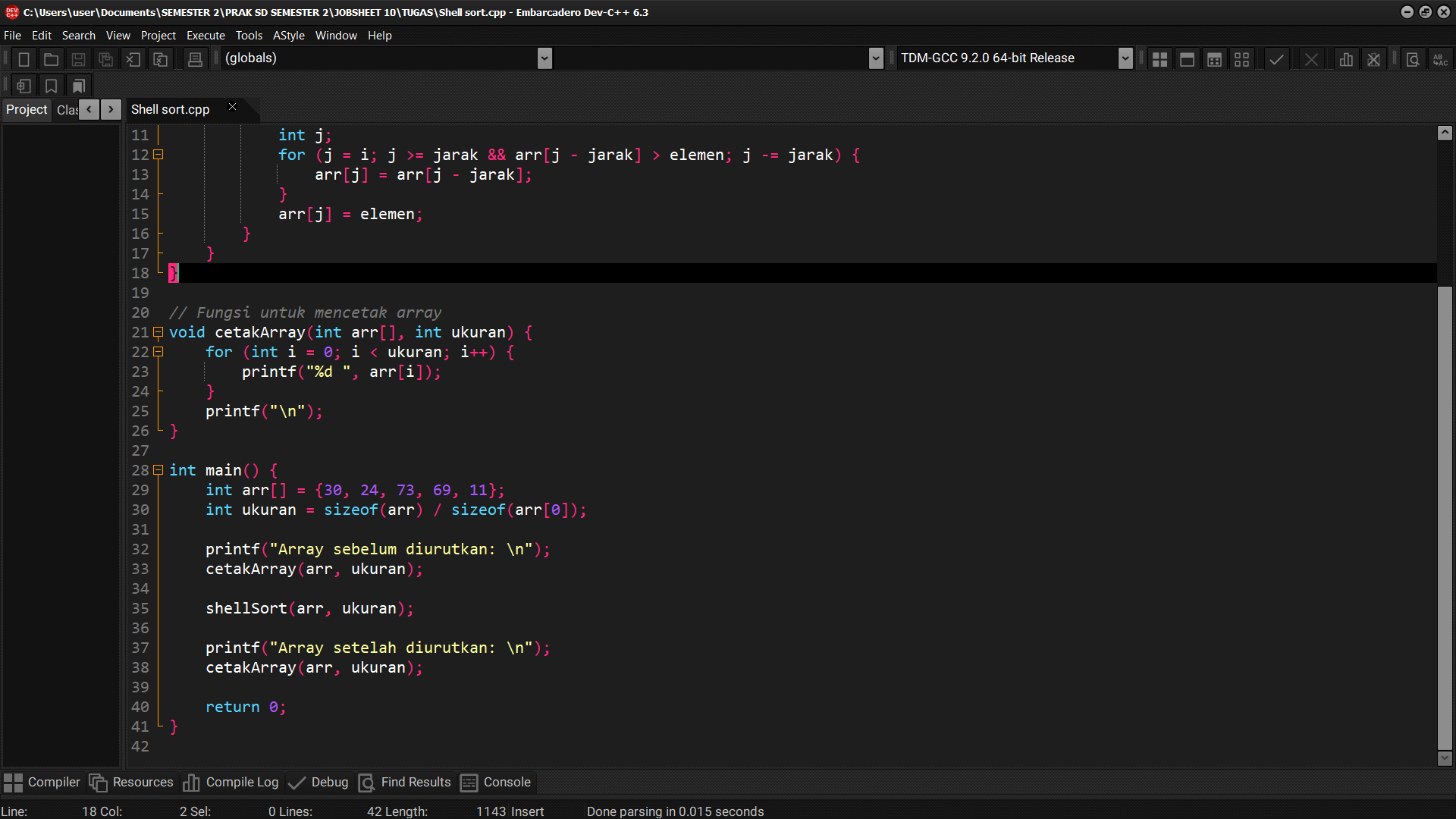
Langkah kedua dengan jarak = 1 (sama seperti Insertion Sort):

Mengurutkan seluruh array: [30, 24, 11, 69, 73] menjadi [11, 24, 30, 69, 73].

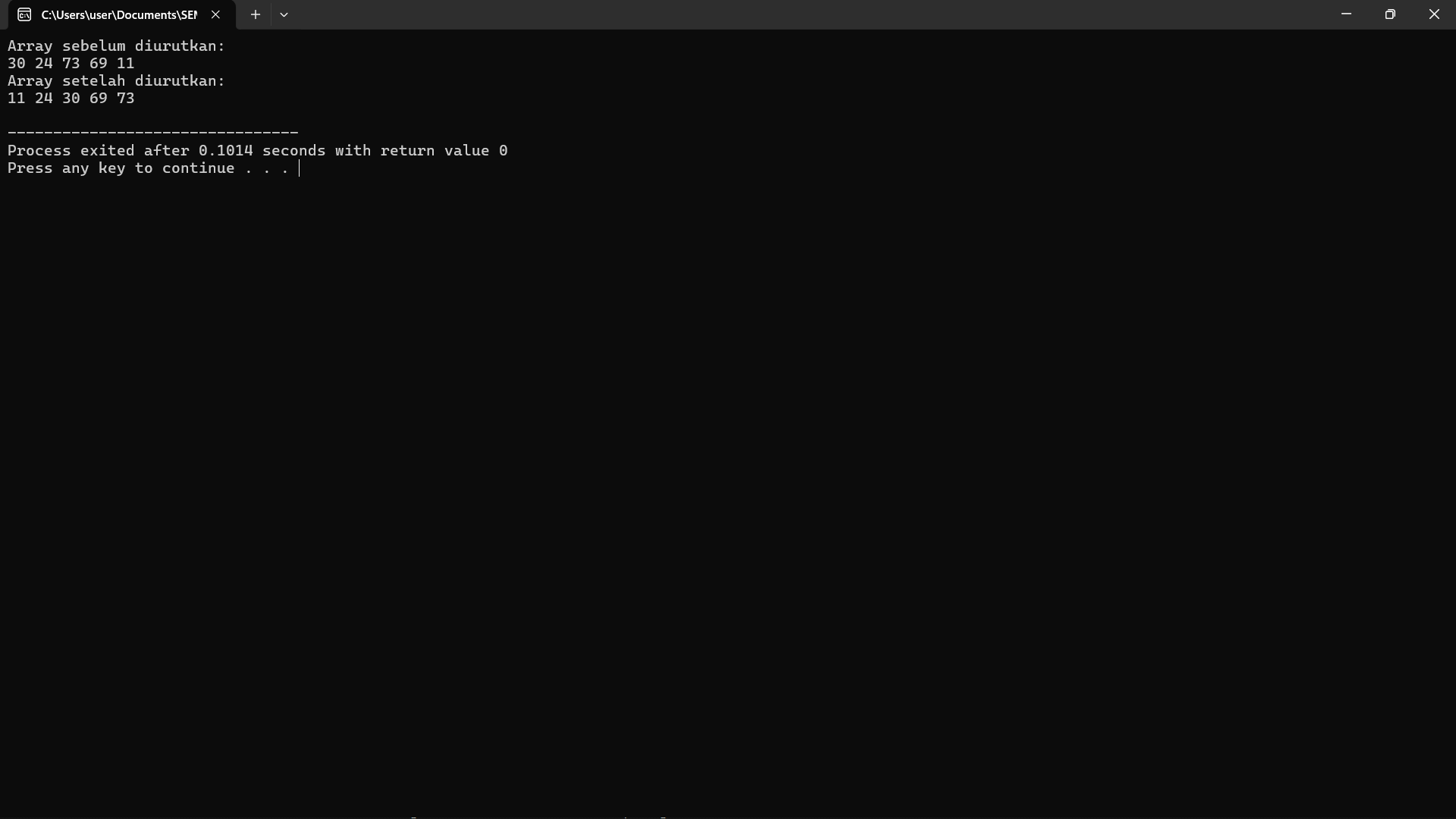
Pada akhirnya, array telah diurutkan.

1. **Screenshot program**





1. **Output**



**Quick Sort**

1. **Source Code**

//Created by 23343079\_Najwa Eldiara Owilia Tiksa

#include <stdio.h>

// Fungsi untuk menukar dua elemen

void tukar(int \*a, int \*b) {

int sementara = \*a;

\*a = \*b;

\*b = sementara;

}

// Fungsi untuk menemukan indeks pivot dan mengatur ulang array

int partisi(int array[], int rendah, int tinggi) {

int pivot = array[tinggi];

int i = (rendah - 1);

for (int j = rendah; j < tinggi; j++) {

if (array[j] <= pivot) {

i++;

tukar(&array[i], &array[j]);

}

}

tukar(&array[i + 1], &array[tinggi]);

return (i + 1);

}

// Fungsi utama Quick Sort

void quickSort(int array[], int rendah, int tinggi) {

if (rendah < tinggi) {

int pi = partisi(array, rendah, tinggi);

quickSort(array, rendah, pi - 1);

quickSort(array, pi + 1, tinggi);

}

}

// Fungsi untuk mencetak array

void cetakArray(int array[], int ukuran) {

for (int i = 0; i < ukuran; i++) {

printf("%d ", array[i]);

}

printf("\n");

}

int main() {

int ukuran;

printf("Masukkan jumlah elemen: ");

scanf("%d", &ukuran);

int array[ukuran];

printf("Masukkan %d elemen:\n", ukuran);

for (int i = 0; i < ukuran; i++) {

scanf("%d", &array[i]);

}

printf("Array sebelum diurutkan: \n");

cetakArray(array, ukuran);

quickSort(array, 0, ukuran - 1);

printf("Array setelah diurutkan: \n");

cetakArray(array, ukuran);

return 0;

}

1. **Penjelasan Program**

Program di atas adalah implementasi dari algoritma pengurutan Quick Sort dalam bahasa C. Quick Sort adalah salah satu algoritma pengurutan yang efisien dan berbasis metode bagi-dan-taklukkan (divide-and-conquer).

Fungsi ini:

Meminta pengguna untuk memasukkan jumlah elemen dalam array.

Mengambil input elemen-elemen array dari pengguna.

Mencetak array sebelum diurutkan.

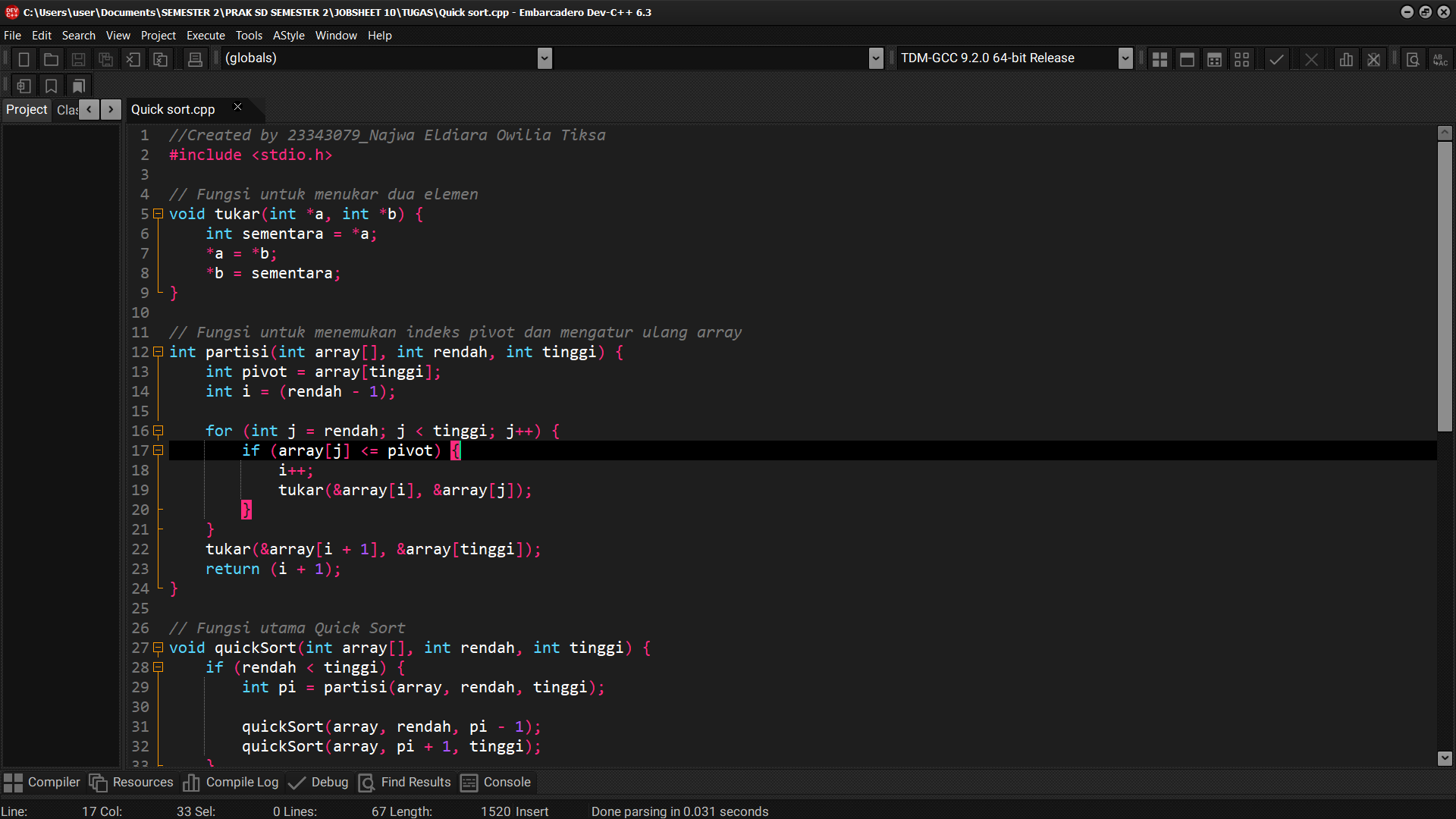
Mengurutkan array menggunakan Quick Sort.

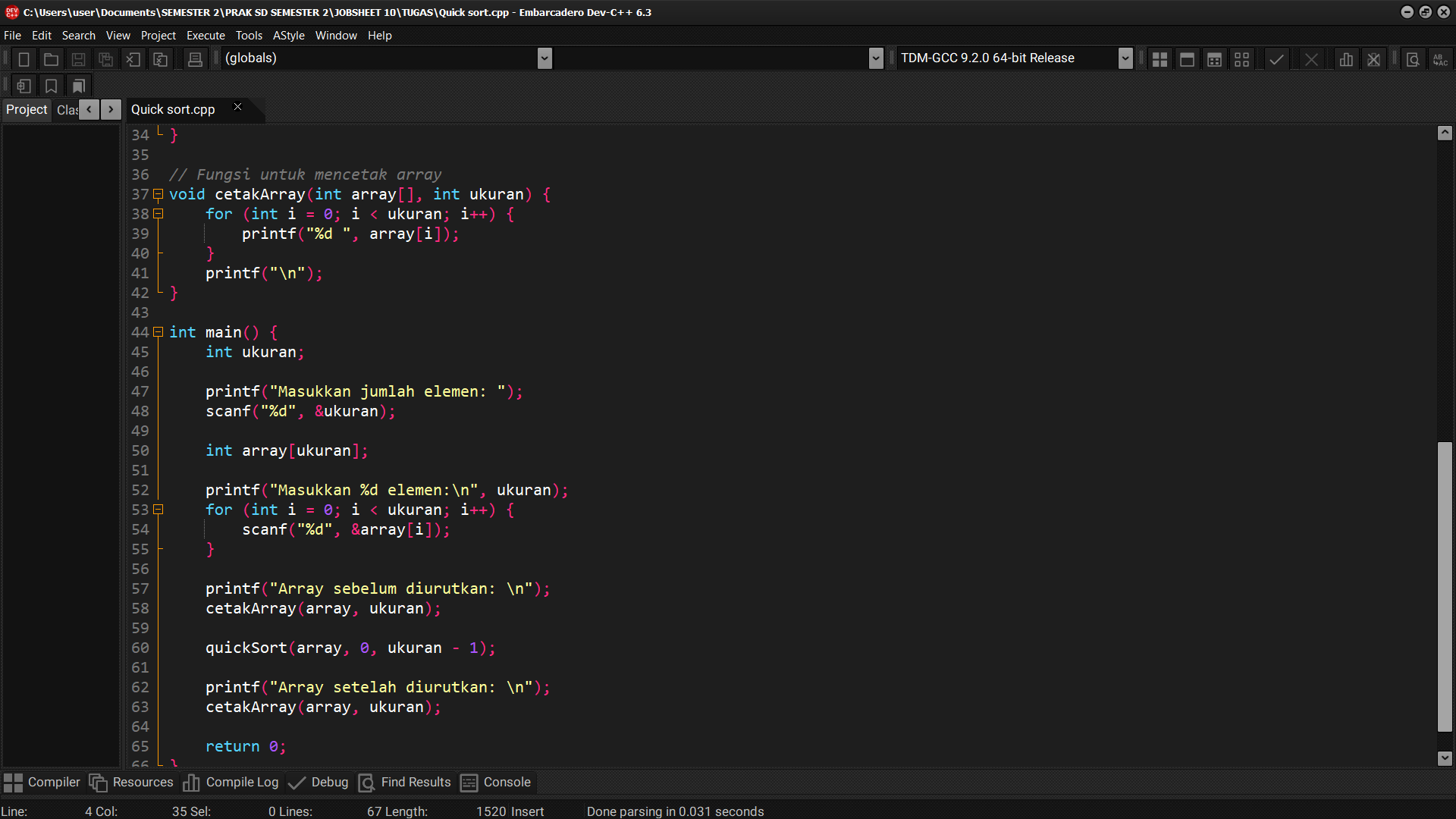
Mencetak array setelah diurutkan.

Algoritma yang Digunakan

Program ini menggunakan algoritma Quick Sort. Quick Sort adalah algoritma pengurutan yang memiliki kompleksitas waktu rata-rata 𝑂 (𝑛 log 𝑛) meskipun dalam kasus terburuk bisa menjadi 𝑂(𝑛2). Quick Sort populer karena biasanya lebih cepat dalam praktik dibandingkan algoritma pengurutan lainnya seperti Merge Sort atau Bubble Sort.

1. **Screenshot program**





1. **Output**

