TUGAS 2 KELAS

RUNNING TIME

Disusun sebagai salah satu tugas

mata kuliah Analisis Algoritma



Tasya Amanda Adinegara – 140810160003

Shofiyyah Nadhiroh – 14018160057

Patricia Joanne – 140810160065

Dikumpulkan tanggal

18 Maret 2019

PROGRAM STUDI S-1 TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

2019

**Soal**

1. Tuliskan algoritma dan running program Menara Hanoi (berapa kali proses pemindahan piringan)!
2. Tuliskan algoritma dan running program bubble sort, selection sort, insertion sort, quick sort, dan merge sort!
3. Analisis running time program-program tersebut dengan mencoba data yang berbeda dan bandingkan satu sama lain. Tuliskan spesifikasi komputer yang digunakan. Tuliskan kesimpulannya!

**Jawaban**

1. Menara Hanoi

Algoritma:

Tower of Hanoi adalah teka-teki matematika di mana kita memiliki tiga batang dan disk. Tujuan dari teka-teki ini adalah untuk memindahkan seluruh tumpukan ke batang lain, mematuhi aturan sederhana berikut:

* + Hanya satu disk yang dapat dipindahkan sekaligus.
  + Setiap gerakan terdiri dari mengambil disk bagian atas dari salah satu tumpukan dan meletakkannya di atas tumpukan lain, yaitu disk hanya dapat dipindahkan jika itu adalah disk paling atas pada tumpukan.
  + Disk tidak dapat ditempatkan di atas disk yang lebih kecil.

Pendekatan:

Ambil contoh untuk 2 disk:

Biarkan batang 1 = 'A', batang 2 = 'B', batang 3 = 'C'.

Langkah 1: Pindahkan disk pertama dari 'A' ke 'B'.

Langkah 2: Pindahkan disk kedua dari 'A' ke 'C'.

Langkah 3: Pindahkan disk pertama dari 'B' ke 'C'.

Polanya di sini adalah:

Pindahkan disk 'n-1' dari 'A' ke 'B'.

Pindahkan disk terakhir dari 'A' ke 'C'.

Pindahkan disk 'n-1' dari 'B' ke 'C'.

Ilustrasi gambar untuk 3 disk:

[](https://media.geeksforgeeks.org/wp-content/uploads/tower-of-hanoi.png)

Contoh:

Input: 2

Output: Disk 1 dipindahkan dari A ke B

Disk 2 dipindahkan dari A ke C

Disk 1 dipindahkan dari B ke C

Input: 3

Output: Disk 1 dipindahkan dari A ke C

Disk 2 dipindahkan dari A ke B

Disk 1 dipindahkan dari C ke B

Disk 3 dipindahkan dari A ke C

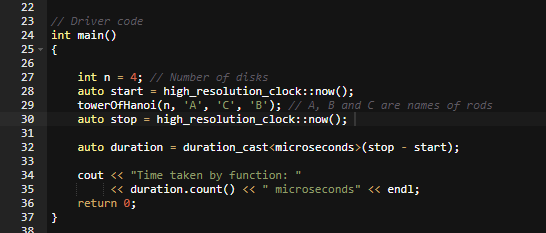
Disk 1 dipindahkan dari B ke A

Disk 2 dipindahkan dari B ke C

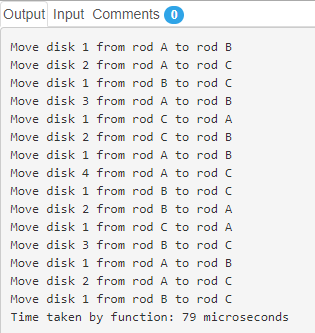
Disk 1 dipindahkan dari A ke C

Program:

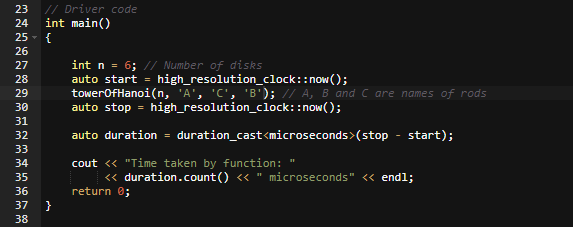




Output:

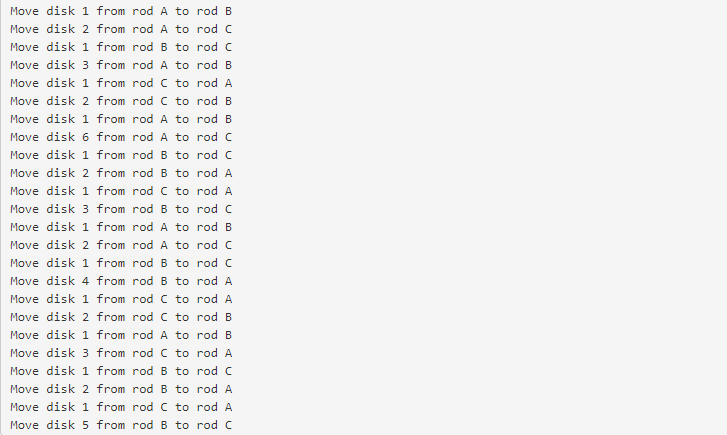


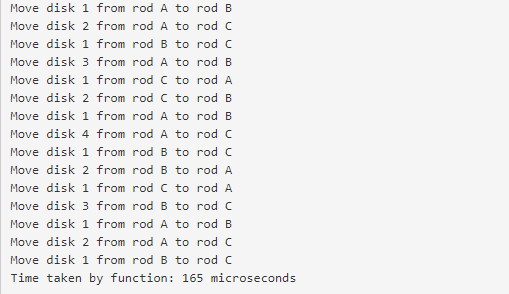
Data yang berbeda:



Output:







1. Bubble Sort

Bubble Sort adalah algoritma pengurutan paling sederhana yang bekerja dengan berulang kali menukar elemen yang berdekatan jika urutannya salah.

Algoritma:  
**Pass Pertama:**

(**5** **1** 4 2 8) 🡪 (**1** **5** 4 2 8)

Di sini, algoritma membandingkan dua elemen pertama, dan bertukar sejak 5 > 1. (1 **5** **4** 2 8) 🡪 (1 **4** **5** 2 8), Tukar sejak 5 > 4

(1 4 **5** **2** 8) 🡪 (1 4 **2** **5** 8), Tukar sejak 5 > 2

(1 4 2 **5** **8**) 🡪 (1 4 2 **5** **8**)

Sekarang, karena elemen-elemen ini sudah berurutan (8 > 5), algoritma tidak menukar mereka.

**Pass Kedua:**

(**1** **4** 2 5 8) 🡪 (**1** **4** 2 5 8)

(1 **4** **2** 5 8) 🡪 (1 **2** **4** 5 8), Tukar sejak 4 > 2

(1 2 **4** **5** 8) 🡪 (1 2 **4** **5** 8)

(1 2 4 **5** **8**) 🡪 (1 2 4 **5** **8**)

Sekarang, array sudah diurutkan, tetapi algoritma ini tidak tahu apakah itu selesai. Algoritma membutuhkan satu pass penuh tanpa swap **apapun** untuk mengetahuinya.

**Pass Ketiga:**

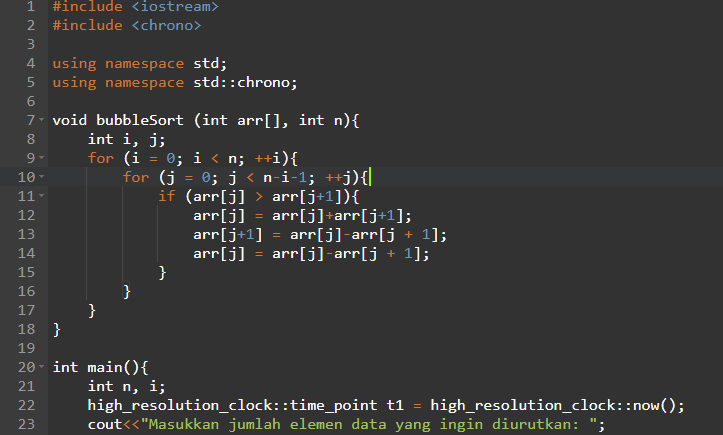
(**1** **2** 4 5 8) 🡪 (**1** **2** 4 5 8)

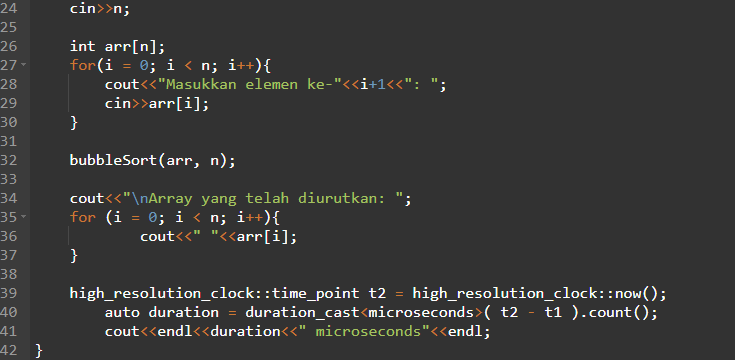
(1 **2** **4** 5 8) 🡪 (1 **2** **4** 5 8)

(1 2 **4** **5** 8) 🡪 (1 2 **4** **5** 8)

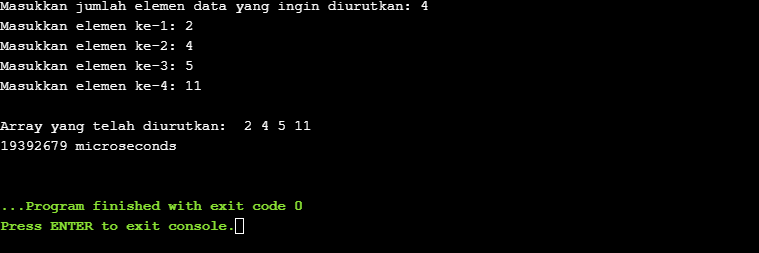
(1 2 4 **5** **8**) 🡪 (1 2 4 **5** **8**)

Program:

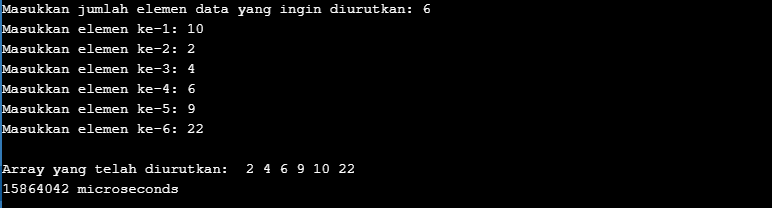




Output:



Data yang berbeda:



1. Selection Sort

Pemilihan algoritma mengurutkan array dengan berulang kali menemukan elemen minimum (mempertimbangkan urutan naik) dari bagian yang tidak disortir dan meletakkannya di awal. Algoritma mempertahankan dua sub-array dalam array yang diberikan.

* + Subarray yang sudah diurutkan.
  + Sisa subarray yang tidak disortir.

Di setiap iterasi dari pemilihan sortir, elemen minimum (mempertimbangkan urutan menaik) dari subarray yang tidak disortir diambil dan dipindahkan ke subarray yang diurutkan.

Algoritma:

arr [] = 64 25 12 22 11

// Temukan elemen minimum di arr [0 ... 4] dan letakkan di awal

**11** 25 12 22 64

// Temukan elemen minimum dalam arr [1 ... 4] dan letakkan di awal arr [1 ... 4]

11 **12** 25 22 64

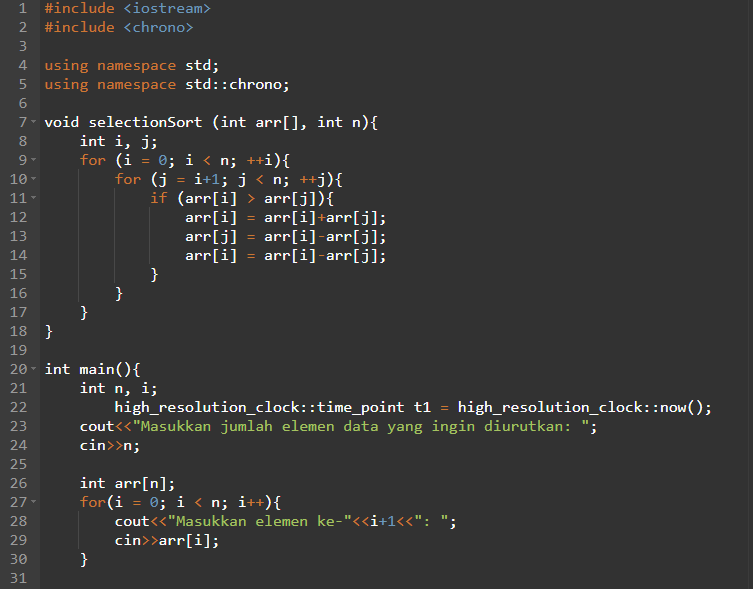
// Temukan elemen minimum dalam arr [2 ... 4] dan letakkan di awal arr [2 ... 4]

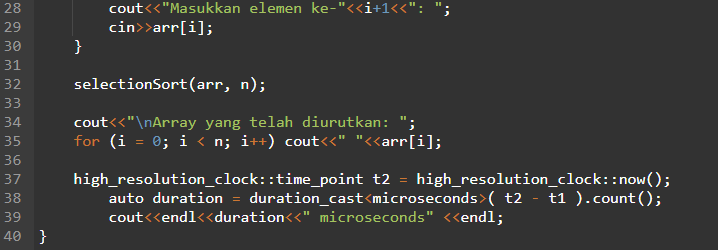
11 12 **22** 25 64

// Temukan elemen minimum dalam arr [3 ... 4] dan letakkan di awal arr [3 ... 4]

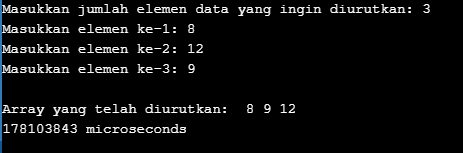
11 12 22 **25** 64

Program:

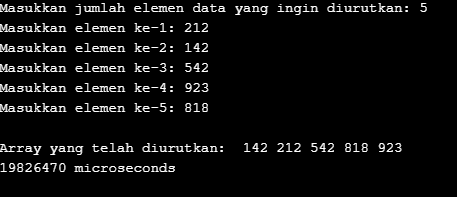




Output:



Data yang berbeda:



1. Insertion Sort

Insertion Sort adalah algoritma penyortiran sederhana yang berfungsi seperti cara kami menyortir kartu remi di tangan kami.

**Algoritma:**  
// Sortir arr [] dengan ukuran n

insertionSort (arr, n)

Loop dari i = 1 ke n-1

Pilih elemen array [i] dan masukkan ke dalam urutan yang diurutkan [0…i-1]

Contoh:

**12**, 11, 13, 5, 6

Mari kita loop untuk i = 1 (elemen kedua array) ke 4 (elemen terakhir array) i = 1. Karena 11 lebih kecil dari 12, pindahkan 12 dan masukkan 11 sebelum 12.  
**11, 12**, 13, 5, 6

i = 2. 13 akan tetap pada posisinya karena semua elemen dalam A [0..I-1] lebih kecil dari 13.

**11, 12, 13**, 5, 6

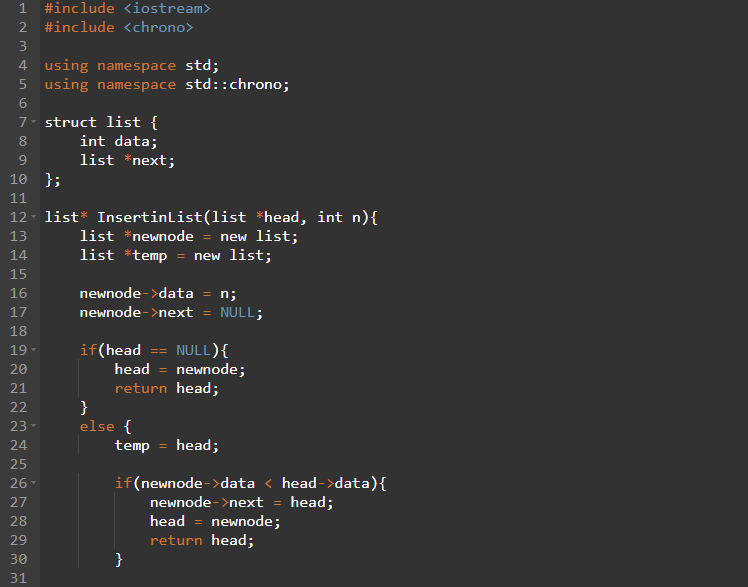
i = 3. 5 akan pindah ke awal dan semua elemen lainnya dari 11 hingga 13 akan bergerak satu posisi di depan posisi mereka saat ini.

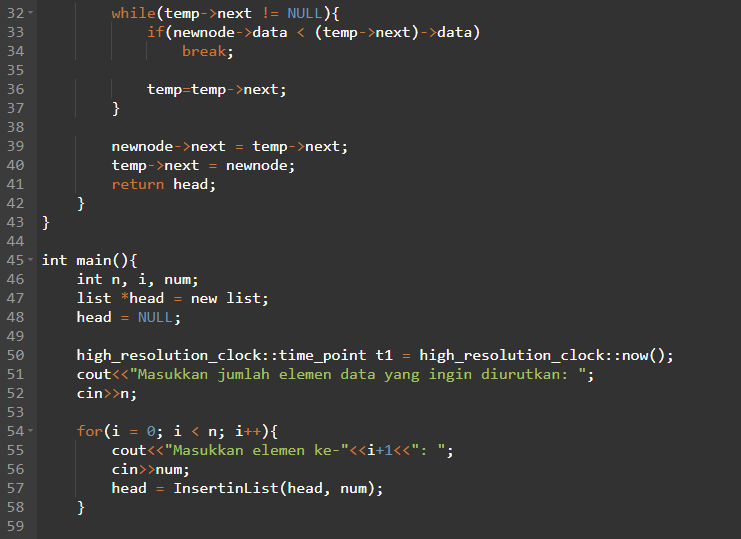
**5, 11, 12, 13** , 6

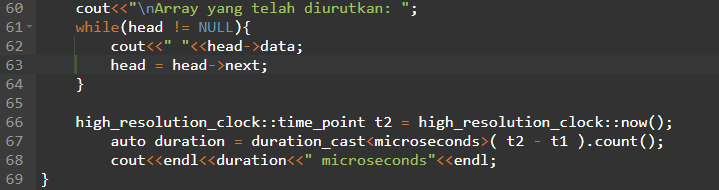
i = 4. 6 akan pindah ke posisi setelah 5, dan elemen dari 11 hingga 13 akan bergerak satu posisi di depan posisi mereka saat ini.

**5, 6, 11, 12, 13**

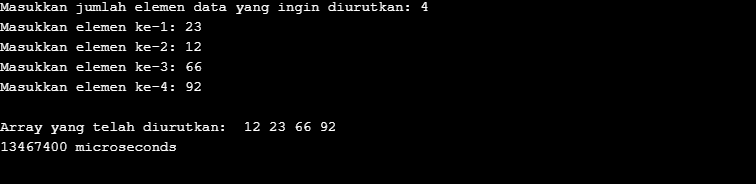
Program:



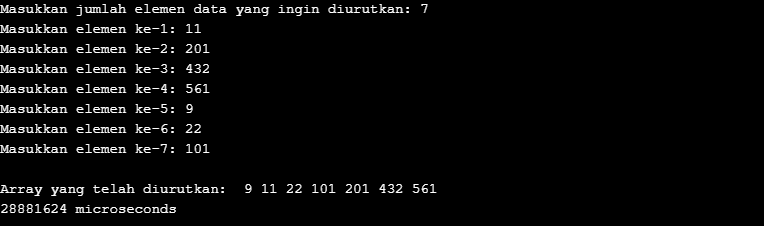




Output:



Data yang berbeda:



1. Quick Sort

Seperti merge sort, quick sort adalah algoritma Divide and Conquer yang mengambil elemen sebagai pivot dan mempartisi array yang diberikan di sekitar pivot yang dipilih. Ada banyak versi quick sort yang memilih pivot dengan berbagai cara.

* Selalu pilih elemen pertama sebagai pivot.
* Selalu pilih elemen terakhir sebagai pivot (diterapkan di bawah)
* Pilih elemen acak sebagai pivot.
* Pilih median sebagai poros.

Proses utama dalam quick sort adalah partisi. Target dari partisi adalah, diberikan array dan elemen x array sebagai pivot, letakkan x pada posisi yang benar dalam array yang diurutkan dan letakkan semua elemen yang lebih kecil (lebih kecil dari x) sebelum x, dan letakkan semua elemen yang lebih besar (lebih besar dari x) setelah x. Semua ini harus dilakukan dalam waktu linier.

Algoritma:

/ \* rendah -> Indeks awal, tinggi -> Indeks akhir \* /

quickSort (arr [], rendah, tinggi){

if (rendah <tinggi){

pi = partisi (arr, rendah, tinggi);

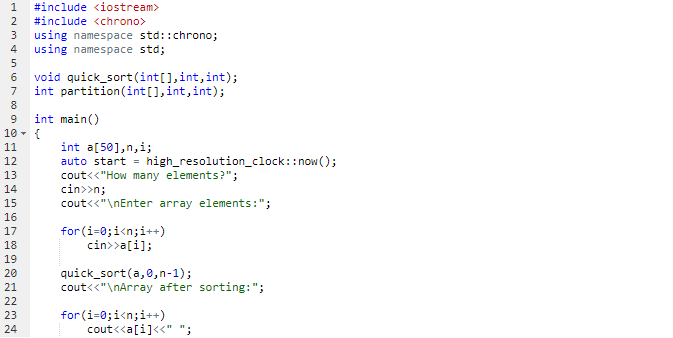
quickSort (arr, low, pi - 1); // Sebelum pi

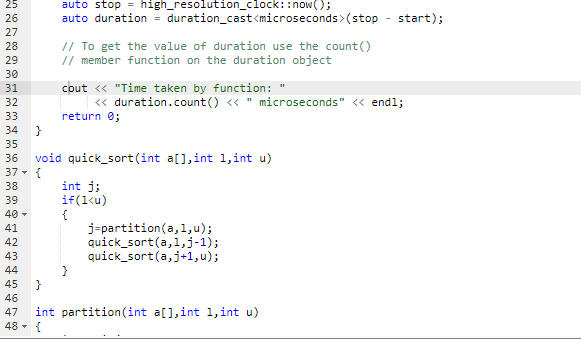
quickSort (arr, pi + 1, tinggi); // Setelah pi

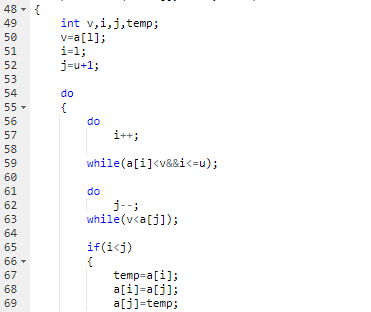
}

}

Program:

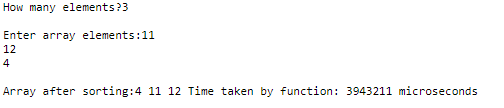




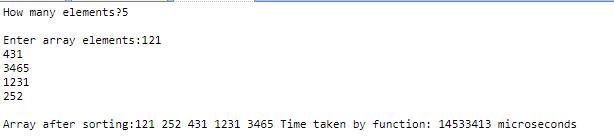




Output:



Data yang berbeda:



1. Merge Sort

Seperti quick sort, merge sort adalah alagoritma Divide and Conquer yang membagi array input dalam dua bagian, memanggil dirinya untuk dua bagian dan kemudian menggabungkan dua bagian yang diurutkan. **Fungsi gabungan()** digunakan untuk menggabungkan dua bagian. Penggabungan (arr, l, m, r) adalah proses utama yang mengasumsikan bahwa arr [l..m] dan arr [m + 1..r] diurutkan dan menggabungkan dua sub-array yang diurutkan menjadi satu.

Algoritma:

Jika r> l

* 1. Temukan titik tengah untuk membagi array menjadi dua bagian:

tengah m = (l + r) / 2

* 1. Panggil mergeSort untuk babak pertama:

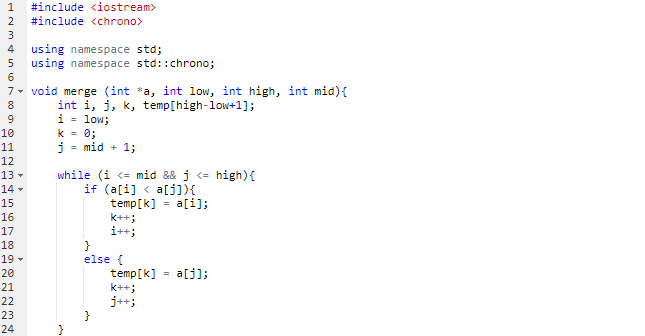
Panggil mergeSort (arr, l, m)

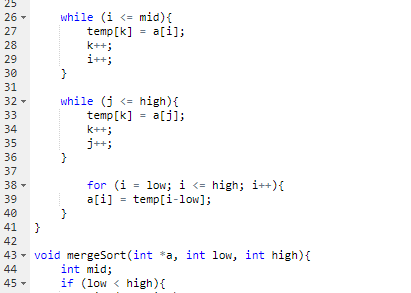
* 1. Panggil mergeSort untuk babak kedua:

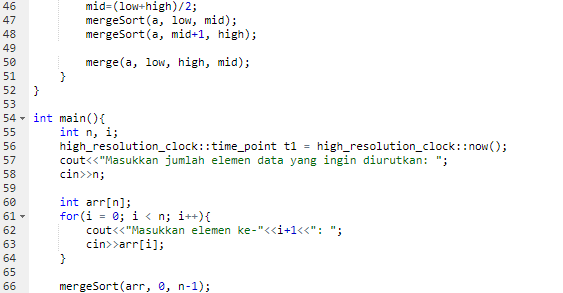
Panggil mergeSort (arr, m + 1, r)

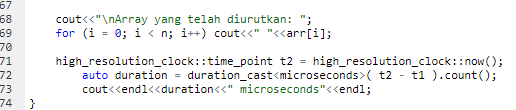
* 1. Gabungkan dua bagian yang diurutkan dalam langkah 2 dan 3:
  2. Panggil gabungan (arr, l, m, r)

Program:

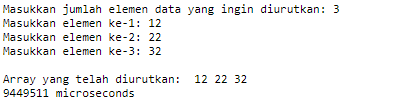




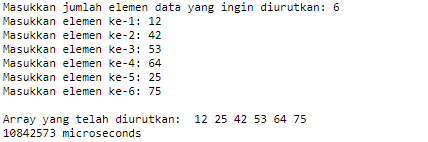




Output:



Data yang berbeda:



**Analisis**

Running time dalam program-program di atas didapatkan dari library chrono pada C++. Running time selalu berubah-ubah bahkan untuk program dengan code dan data yang sama sekalipun. Running time dipengaruhi oleh kerja CPU dan kita tidak pernah tahu apa saja proses yang sedang dikerjakan oleh CPU kita. Untuk spesifikasi dari PC yang digunakan untuk menjalankan program-program di atas adalah sebagai berikut:

