**Laporan Resmi**

**Praktikum Algoritma dan Struktur Data**

**Insertion and Selection Sort**



**Dr. Tita Karlita S.Kom, M.Kom**

**Nama : Marits Ikmal Yasin**

**Kelas : 1D4 IT B**

**NRP : 3121600047**

1. Buatlah program yang mengimplementasikan algoritma selection sort, untuk mengurutkan 10 buah elemen array bertipe integer. Tampilkan hasil pengurutan data dalam urutan ascending. Tampilkan jumlah perbandingan dan jumlah penukaran elemen. Gunakan data berikut ini dalam 4 kali running program untuk melihat efek dari distribusi elemen terhadap jumlah penukaran dan jumlah perbandingan.
2. array 1: 3, 10, 4, 6, 8, 9, 7, 2, 1, 5 -- data awal, semua elemen terdistribusi acak
3. array 2: 1, 2, 3, 4, 5, 9, 7, 6, 8, 10-- data awal, 5 elemen pertama terurut descending, dan 5 elemen terakhir acak
4. array 3: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 -- data awal, semua elemen terurut ascending
5. array 4: 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 -- data awal, semua elemen terurut descending

Kode :

#include <stdio.h>

#include <stdlib.h>

#include <sys/time.h>

#include <unistd.h>

int perbandingan, penukaran;

void printArray(int arr[], int sz)

{

for (int i = 0; i < sz; i++)

{

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

}

printf("\n");

}

void selectionSort(int arr[], int sz)

{

for (int step = 0; step < sz - 1; step++)

{

usleep(1000);

int min = step;

for (int i = step + 1; i < sz; i++)

{

perbandingan++;

if (arr[i] < arr[min])

{

min = i;

}

}

if (arr[min] != arr[step])

penukaran++;

int temp = arr[min];

arr[min] = arr[step];

arr[step] = temp;

}

}

int main()

{

struct timeval start, stop;

/\*a\*/int array[10] = {3, 10, 4, 6, 8, 9, 7, 2, 1, 5};

/\*b\*/ int array[10] = {1, 2, 3, 4, 5, 9, 7, 6, 8, 10};

/\*c\*/ int array[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

/\*d\*/ int array[10] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1};

int size = sizeof(array) / sizeof(array[0]);

puts("Data Sebelum Sorting");

printArray(array, size);

gettimeofday(&start, NULL);

selectionSort(array, size);

gettimeofday(&stop, NULL);

puts("Data Setelah Sorting");

printArray(array, size);

printf("Sorting %lu us\n", ((stop.tv\_sec - start.tv\_sec) \* 1000000 + stop.tv\_usec - start.tv\_usec));

printf("Perbandingan = %d\n", perbandingan);

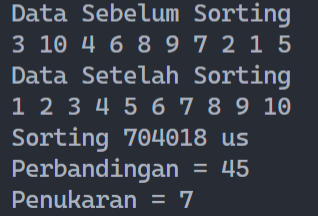
printf("Penukaran = %d\n", penukaran);

return 0;

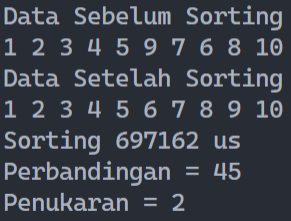
}

Output :

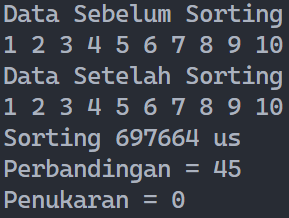
1. int array[10] = {3, 10, 4, 6, 8, 9, 7, 2, 1, 5};



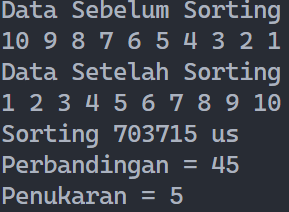
1. int array[10] = {1, 2, 3, 4, 5, 9, 7, 6, 8, 10};



1. int array[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};



1. int array[10] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1};



1. Buatlah program yang mengimplementasikan algoritma insertion sort, untuk mengurutkan 10 buah elemen array bertipe integer. Tampilkan hasil pengurutan data dalam urutan ascending. Tampilkan jumlah perbandingan dan jumlah penukaran elemen. Gunakan data berikut ini dalam 4 kali running program untuk melihat efek dari distribusi elemen terhadap jumlah pergeseran, jumlah perbandingan, dan jumlah penyisipan.
2. array 1: 3, 10, 4, 6, 8, 9, 7, 2, 1, 5 -- data awal, semua elemen terdistribusi acak
3. array 2: 1, 2, 3, 4, 5, 9, 7, 6, 8, 10-- data awal, 5 elemen pertama terurut descending, dan 5 elemen terakhir acak
4. array 3: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 -- data awal, semua elemen terurut ascending
5. array 4: 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 -- data awal, semua elemen terurut descending

Kode :

#include <stdio.h>

#include <stdlib.h>

#include <sys/time.h>

#include <unistd.h>

int perbandingan, pergeseran, penyisipan;

void printArray(int arr[], int sz)

{

for (int i = 0; i < sz; i++)

{

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

}

printf("\n");

}

void insertionSort(int arr[], int sz)

{

for (int step = 1; step < sz; step++)

{

int key = arr[step];

int back = step - 1;

// best case comparison is n-1

if (key > arr[back])

{

perbandingan++;

usleep(1000);

}

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

{

// worst case comparison is n(n-1)/2

usleep(1000);

pergeseran++;

arr[back + 1] = arr[back];

back--;

perbandingan++;

}

if (arr[back + 1] != key)

penyisipan++;

arr[back + 1] = key;

}

}

int main()

{

struct timeval start, stop;

/\*a\*/ int array[10] = {3, 10, 4, 6, 8, 9, 7, 2, 1, 5};

/\*b\*/ int array[10] = {1, 2, 3, 4, 5, 9, 7, 6, 8, 10};

/\*c\*/ int array[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

/\*d\*/ int array[10] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1};

int size = sizeof(array) / sizeof(array[0]);

puts("Data Sebelum Sorting");

printArray(array, size);

gettimeofday(&start, NULL);

insertionSort(array, size);

gettimeofday(&stop, NULL);

puts("Data Setelah Sorting");

printArray(array, size);

printf("Sorting %lu us\n", ((stop.tv\_sec - start.tv\_sec) \* 1000000 + stop.tv\_usec - start.tv\_usec));

printf("Perbandingan = %d\n", perbandingan);

printf("Pergeseran = %d\n", pergeseran);

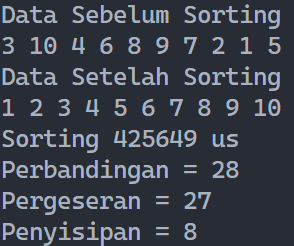
printf("Penyisipan = %d\n", penyisipan);

return 0;

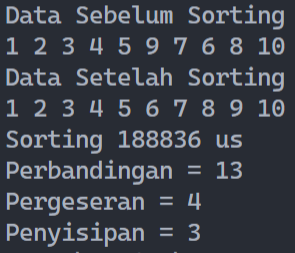
}

Output :

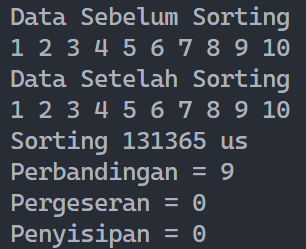
1. int array[10] = {3, 10, 4, 6, 8, 9, 7, 2, 1, 5};



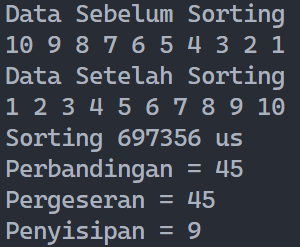
1. int array[10] = {1, 2, 3, 4, 5, 9, 7, 6, 8, 10};



1. int array[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};



1. int array[10] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1};



1. Buatlah program yang mengimplementasikan algoritma insertion dan selection sort, untuk mengurutkan 10 buah elemen array bertipe integer dengan menggunakan array berikut sebagai inisialisasi nilia awal array: 3, 10, 4, 6, 8, 9, 7, 2, 1, 5

Kode :

#include <stdio.h>

#include <stdlib.h>

#include <sys/time.h>

#include <unistd.h>

int perbandingan, penukaran;

void printArray(int arr[], int sz)

{

for (int i = 0; i < sz; i++)

{

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

}

printf("\n");

}

// Ascending

void AscSelectionSort(int arr[], int sz)

{

for (int step = 0; step < sz - 1; step++)

{

int min = step;

for (int i = step + 1; i < sz; i++)

{

usleep(100);

perbandingan++;

if (arr[i] < arr[min])

min = i;

}

if (arr[min] != arr[step])

penukaran++;

int temp = arr[min];

arr[min] = arr[step];

arr[step] = temp;

}

}

// Descending

void DescSelectionSort(int arr[], int sz)

{

for (int step = 0; step < sz - 1; step++)

{

int min = step;

for (int i = step + 1; i < sz; i++)

{

usleep(100);

perbandingan++;

if (arr[i] > arr[min])

min = i;

}

if (arr[min] != arr[step])

penukaran++;

int temp = arr[min];

arr[min] = arr[step];

arr[step] = temp;

}

}

// Ascending

void AscInsertionSort(int arr[], int sz)

{

for (int step = 1; step < sz; step++)

{

int key = arr[step];

int back = step - 1;

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

{

usleep(100);

arr[back + 1] = arr[back];

back--;

}

if (key > arr[back])

{

perbandingan++;

usleep(100);

}

if (arr[back + 1] != key)

penukaran++;

arr[back + 1] = key;

}

}

// Descending

void DescInsertionSort(int arr[], int sz)

{

for (int step = 1; step < sz; step++)

{

int key = arr[step];

int back = step - 1;

while (key > arr[back] && back >= 0)

{

usleep(100);

arr[back + 1] = arr[back];

back--;

}

if (key < arr[back])

{

perbandingan++;

usleep(100);

}

if (arr[back + 1] != key)

penukaran++;

arr[back + 1] = key;

}

}

void menu()

{

puts("MENU METODE SORTING");

puts("1. Insertion Sort");

puts("2. Selection Sort");

puts("3. Keluar");

}

void insertion()

{

puts("MENU METODE SORTING");

puts("1. Ascending");

puts("2. Descending");

}

void selection()

{

puts("MENU METODE SORTING");

puts("1. Ascending");

puts("2. Descending");

}

int main()

{

struct timeval start, stop;

int array[10] = {3, 10, 4, 6, 8, 9, 7, 2, 1, 5};

int size = sizeof(array) / sizeof(array[0]);

int pilihan, pilih;

printArray(array, size);

menu();

printf("Pilihan Anda [1/2/3] : ");

scanf("%d", &pilihan);

switch (pilihan)

{

case 1:

insertion();

break;

case 2:

selection();

break;

case 3:

exit(0);

default:

exit(0);

}

printf("Pilihan Anda [1/2] : ");

scanf("%d", &pilih);

gettimeofday(&start, NULL);

if (pilihan == 1)

{

switch (pilih)

{

case 1:

AscInsertionSort(array, size);

break;

case 2:

DescInsertionSort(array, size);

break;

default:

exit(0);

}

}

else

{

switch (pilih)

{

case 1:

AscSelectionSort(array, size);

break;

case 2:

DescSelectionSort(array, size);

break;

default:

exit(0);

}

}

gettimeofday(&stop, NULL);

printf("Perbandingan = %d\n", perbandingan);

printf("Penukaran = %d\n", penukaran);

printf("Sorting %lu us\n", ((stop.tv\_sec - start.tv\_sec) \* 1000000 + stop.tv\_usec - start.tv\_usec));

printArray(array, size);

return 0;

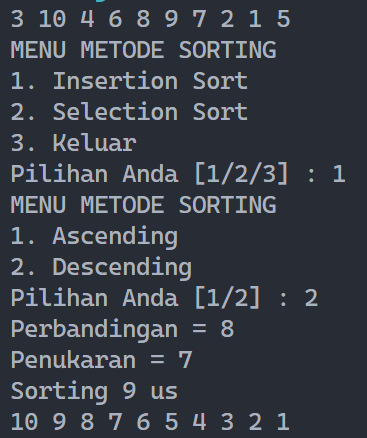
}

Output :

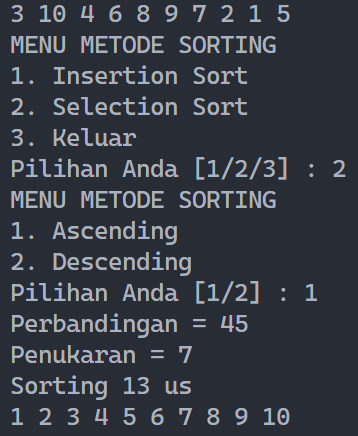
1. Insertion
   * Ascending



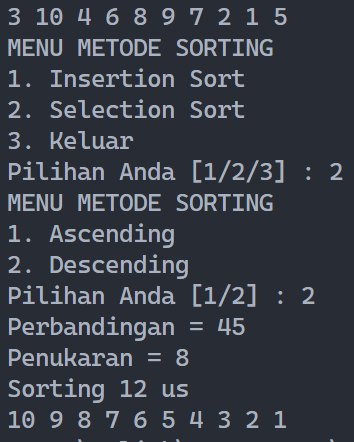
* + Descending



1. Selection
   * Ascending



* + Descending



1. 1000 data Array generate secara random. Kemudian diurutkan menggunakan Insertion dan juga Selection Sort
2. Insertion Sort

Kode :

#include <stdio.h>

#include <stdlib.h>

#include <sys/time.h>

#include <unistd.h>

#define MAKS 1000

int perbandingan, pergeseran, penyisipan;

void printArray(int arr[], int sz)

{

for (int i = 0; i < sz; i++)

{

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

}

printf("\n");

}

void insertionSort(int arr[], int sz)

{

for (int step = 1; step < sz; step++)

{

int key = arr[step];

int back = step - 1;

// best case comparison is n-1

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

{

// worst case comparison is n(n-1)/2

usleep(100);

pergeseran++;

arr[back + 1] = arr[back];

back--;

perbandingan++;

}

if (key > arr[back])

{

perbandingan++;

usleep(100);

}

if (arr[back + 1] != key)

penyisipan++;

arr[back + 1] = key;

}

}

int main()

{

struct timeval start, stop;

int array[MAKS];

for (int i = 0; i < MAKS; i++)

{

array[i] = rand();

}

int size = sizeof(array) / sizeof(array[0]);

puts("Data Sebelum Sorting");

printArray(array, size);

gettimeofday(&start, NULL);

insertionSort(array, size);

gettimeofday(&stop, NULL);

puts("Data Setelah Sorting");

printArray(array, size);

printf("Sorting %lu us\n", ((stop.tv\_sec - start.tv\_sec) \* 1000000 + stop.tv\_usec - start.tv\_usec));

printf("Perbandingan = %d\n", perbandingan);

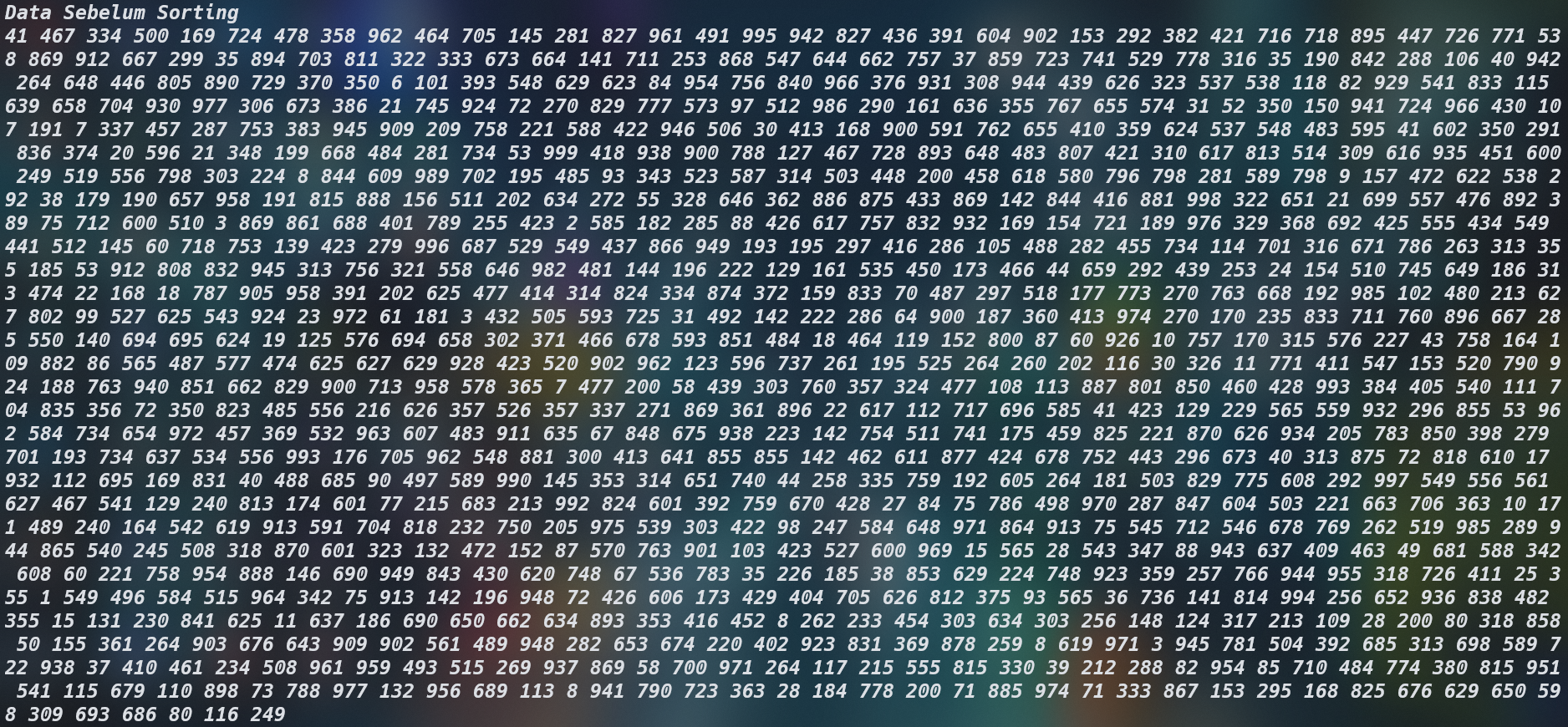
printf("Pergeseran = %d\n", pergeseran);

printf("Penyisipan = %d\n", penyisipan);

return 0;

}

Output :





1. Selection Sort

Kode :

#include <stdio.h>

#include <stdlib.h>

#include <sys/time.h>

#include <unistd.h>

#define MAKS 1000

int perbandingan, penukaran;

void printArray(int arr[], int sz)

{

for (int i = 0; i < sz; i++)

{

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

}

printf("\n");

}

void selectionSort(int arr[], int sz)

{

for (int step = 0; step < sz - 1; step++)

{

int min = step;

for (int i = step + 1; i < sz; i++)

{

usleep(100);

perbandingan++;

if (arr[i] < arr[min])

{

min = i;

}

}

if (arr[min] != arr[step])

penukaran++;

int temp = arr[min];

arr[min] = arr[step];

arr[step] = temp;

}

}

int main()

{

struct timeval start, stop;

int array[MAKS];

for (int i = 0; i < MAKS; i++)

{

array[i] = rand() % 1000;

}

int size = sizeof(array) / sizeof(array[0]);

puts("Data Sebelum Sorting");

printArray(array, size);

gettimeofday(&start, NULL);

selectionSort(array, size);

gettimeofday(&stop, NULL);

puts("Data Setelah Sorting");

printArray(array, size);

printf("Sorting %lu us\n", ((stop.tv\_sec - start.tv\_sec) \* 1000000 + stop.tv\_usec - start.tv\_usec));

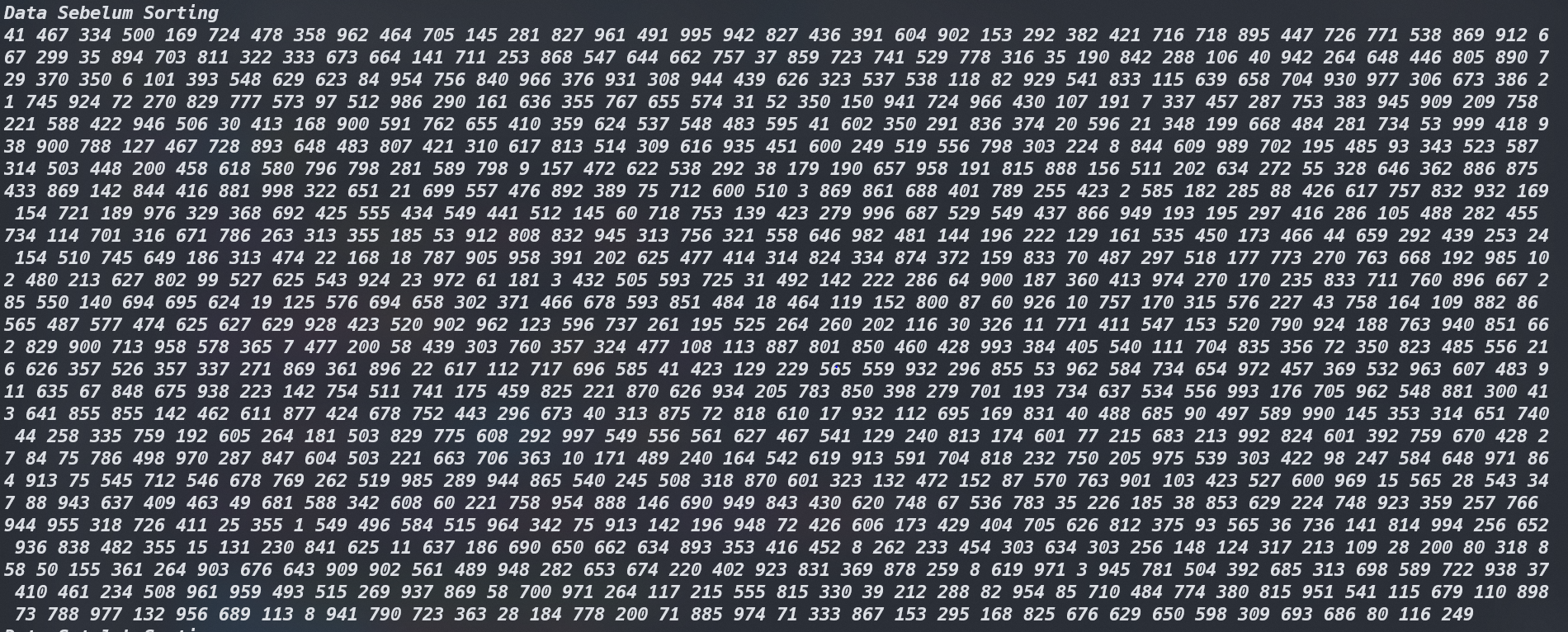
printf("Perbandingan = %d\n", perbandingan);

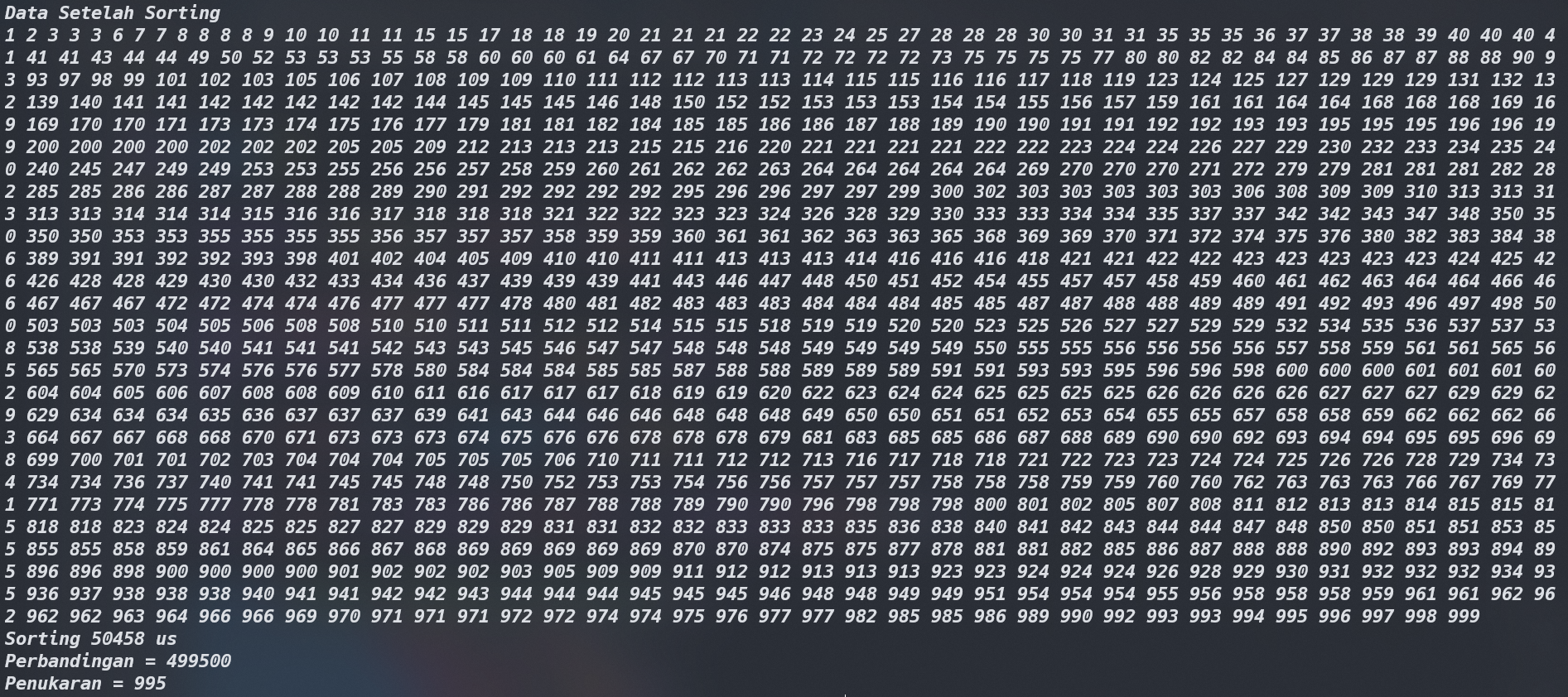
printf("Penukaran = %d\n", penukaran);

return 0;

}

Output :





Analisa :

1 ) Selection Sort

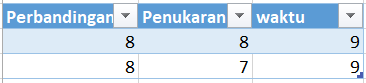
|  |  |  |
| --- | --- | --- |
| **Perbandingan** | **Penukaran** | **Waktu** |
| 45 | 7 | 50 |
| 45 | 2 | 45 |
| 45 | 0 | 45.36610268 |
| 45 | 5 | 49.77902567 |

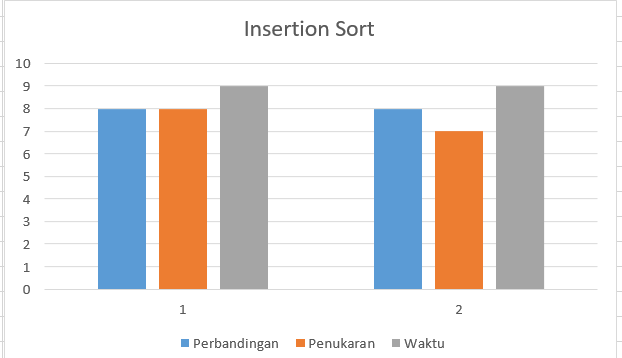
2) Insertion Sort

|  |  |  |  |
| --- | --- | --- | --- |
| **Perbandingan** | **Pergeseran** | **Penukaran** | **Waktu** |
| 28 | 27 | 8 | 37.99861659 |
| 13 | 4 | 3 | 27.5385121 |
| 9 | 0 | 0 | 25 |
| 45 | 45 | 9 | 50 |

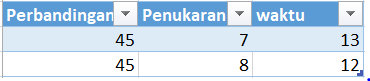
Menu

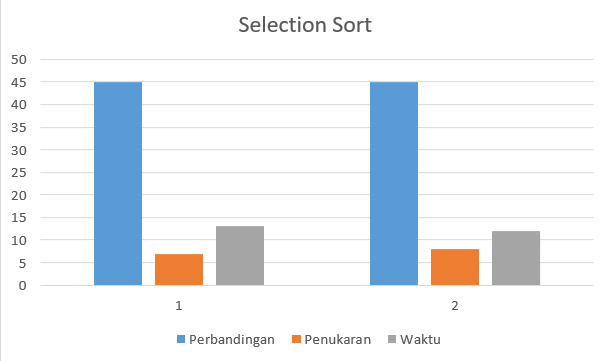
1. Insertion Sort





1. Selection Sort





Generete nilai random



Kesimpulan

Waktu kompleksitas pada Selection sort pada semua kasus yaitu O(n2). Sedangkan Insertion sort antara best case dan worst casenya berbeda. Insertion sort best case memiliki waktu kompleksitas O(n) dan untuk worst casenya O(n2). Jadi apabila membicarakan mana yang lebih cepat antara Selection sort dan Insertion sort maka insertion akan lebih cepat karena lebih stabil. Semakin sedikit perbandingan yang dilakukan maka semakin cepat pula proses pengurutan. Beda halnya dengan selection sort. Apabila data banyak mau sudah terurut atau belum maka kecepatannya sama. Hal ini didasari oleh waktu komplesitas di atas.