**Kode Percobaan & Latihan dalam Bentuk WORD**

Praktikum Algoritma dan Struktur Data

****

**Oleh:**

Samudero Dirgantoro / 5223600016

**Program Studi D4 Teknologi Game**

**Departemen Teknologi Multimedia Kreatif**

**Politeknik Elektronika Negeri Surabaya**

**2023/2024**

1. Percobaan:

//Samudero Dirgantoro 5223600016

#include <iostream>

#include <cstdlib>

#include <ctime>

#define MAX 10

int Data[MAX];

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

int temp = \*a;

\*a = \*b;

\*b = temp;

}

// Pengurutan Quick rekursif

void QuickRekursif(int kiri, int kanan) {

if (kiri < kanan) {

int pivot = kiri;

int i = kiri;

int j = kanan;

while (i < j) {

while (Data[i] <= Data[pivot] && i <= kanan)

i++;

while (Data[j] > Data[pivot] && j >= kiri)

j--;

if (i < j)

Tukar(&Data[i], &Data[j]);

}

Tukar(&Data[j], &Data[pivot]);

QuickRekursif(kiri, j - 1);

QuickRekursif(j + 1, kanan);

}

}

// Pengurutan Quick non-rekursif

void Quick() {

int kiri = 0;

int kanan = MAX - 1;

int stack[MAX];

int top = -1;

stack[++top] = kiri;

stack[++top] = kanan;

while (top >= 0) {

kanan = stack[top--];

kiri = stack[top--];

int pivot = kiri;

int i = kiri;

int j = kanan;

while (i < j) {

while (Data[i] <= Data[pivot] && i <= kanan)

i++;

while (Data[j] > Data[pivot] && j >= kiri)

j--;

if (i < j)

Tukar(&Data[i], &Data[j]);

}

Tukar(&Data[j], &Data[pivot]);

if (kiri < j - 1) {

stack[++top] = kiri;

stack[++top] = j - 1;

}

if (j + 1 < kanan) {

stack[++top] = j + 1;

stack[++top] = kanan;

}

}

}

// Pengurutan Merge

void Merge(int Data[], int temp[], int kiri, int middle, int kanan) {

int i = kiri;

int j = middle + 1;

int k = kiri;

while (i <= middle && j <= kanan) {

if (Data[i] <= Data[j])

temp[k++] = Data[i++];

else

temp[k++] = Data[j++];

}

while (i <= middle)

temp[k++] = Data[i++];

while (j <= kanan)

temp[k++] = Data[j++];

for (i = kiri; i <= kanan; i++)

Data[i] = temp[i];

}

void MergeSort(int Data[], int temp[], int kiri, int kanan) {

if (kiri < kanan) {

int middle = (kiri + kanan) / 2;

MergeSort(Data, temp, kiri, middle);

MergeSort(Data, temp, middle + 1, kanan);

Merge(Data, temp, kiri, middle, kanan);

}

}

void TampilkanDataSebelum(const char\* label) {

std::cout << label << "\n";

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

std::cout << "Data sebelum ke " << i << " : " << Data[i] << "\n";

}

}

void TampilkanDataSesudah(const char\* label) {

std::cout << label << "\n";

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

std::cout << "Data sesudah ke " << i << " : " << Data[i] << "\n";

}

}

int main() {

int pilihan;

srand(time(0));

std::cout << "Membangkitkan bilangan acak\n";

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

Data[i] = rand() % 1000 + 1;

}

std::cout << "Pilih metode pengurutan:\n1. Quick Sort Rekursif\n2. Quick Sort Non-Rekursif\n3. Merge Sort\n";

std::cout << "Input: ";

std::cin >> pilihan;

// Tampilkan data sebelum diurutkan

TampilkanDataSebelum("Data Sebelum Diurutkan:");

// Menu

switch (pilihan) {

case 1:

QuickRekursif(0, MAX - 1);

TampilkanDataSesudah("Setelah diurut dengan Quick Rekursif:");

break;

case 2:

Quick();

TampilkanDataSesudah("Setelah diurut dengan Quick Non-Rekursif:");

break;

case 3:

int temp[MAX];

MergeSort(Data, temp, 0, MAX - 1);

TampilkanDataSesudah("Setelah diurut dengan Merge:");

break;

default:

std::cout << "Error";

break;

}

}

1. Latihan:

//Samudero Dirgantoro 5223600016

//Di Visual Studio ada error dan tidak bisa dijalankan. Tetapi di Programiz bisa.

#include <iostream>

#include <string>

#include <vector>

using namespace std;

struct Pegawai {

int NIP;

string Nama;

string Alamat;

char Golongan;

};

int NIP(Pegawai arr[], int low, int high, bool naik, int& perbandingan);

int Nama(Pegawai arr[], int low, int high, bool naik, int& perbandingan);

void merge(Pegawai arr[], int low, int mid, int high, bool naik, bool berdasarkanNIP, int& perbandingan, int& pergeseran);

void tampilkanPegawai(Pegawai arr[], int n) {

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

cout << "NIP: " << arr[i].NIP << ", Nama: " << arr[i].Nama << ", Alamat: " << arr[i].Alamat << ", Golongan: " << arr[i].Golongan << "\n";

}

cout << "\n";

}

void Tukar(Pegawai& a, Pegawai& b) {

Pegawai temp = a;

a = b;

b = temp;

}

void quickSortRekursif(Pegawai arr[], int low, int high, bool naik, bool berdasarkanNIP, int& perbandingan, int& pergeseran) {

if (low < high) {

int pi;

if (berdasarkanNIP)

pi = NIP(arr, low, high, naik, perbandingan);

else

pi = Nama(arr, low, high, naik, perbandingan);

quickSortRekursif(arr, low, pi - 1, naik, berdasarkanNIP, perbandingan, pergeseran);

quickSortRekursif(arr, pi + 1, high, naik, berdasarkanNIP, perbandingan, pergeseran);

}

}

void quickSortNonRekursif(Pegawai arr[], int low, int high, bool naik, bool berdasarkanNIP, int& perbandingan, int& pergeseran) {

vector<int> stack(high - low + 1);

int top = -1;

stack[++top] = low;

stack[++top] = high;

while (top >= 0) {

high = stack[top--];

low = stack[top--];

int pi;

if (berdasarkanNIP)

pi = NIP(arr, low, high, naik, perbandingan);

else

pi = Nama(arr, low, high, naik, perbandingan);

if (pi - 1 > low) {

stack[++top] = low;

stack[++top] = pi - 1;

}

if (pi + 1 < high) {

stack[++top] = pi + 1;

stack[++top] = high;

}

}

}

// Pengurutan Merge

void merge(Pegawai arr[], int low, int high, bool naik, bool berdasarkanNIP, int& perbandingan, int& pergeseran) {

if (low < high) {

int mid = low + (high - low) / 2;

merge(arr, low, mid, naik, berdasarkanNIP, perbandingan, pergeseran);

merge(arr, mid + 1, high, naik, berdasarkanNIP, perbandingan, pergeseran);

merge(arr, low, mid, high, naik, berdasarkanNIP, perbandingan, pergeseran);

}

}

//gabung

void merge(Pegawai arr[], int low, int mid, int high, bool naik, bool berdasarkanNIP, int& perbandingan, int& pergeseran) {

int n1 = mid - low + 1;

int n2 = high - mid;

// Kiri kanan merge

Pegawai L[n1], R[n2];

for (int i = 0; i < n1; i++)

L[i] = arr[low + i];

for (int j = 0; j < n2; j++)

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

int i = 0, j = 0, k = low;

while (i < n1 && j < n2) {

if ((berdasarkanNIP && ((naik && L[i].NIP <= R[j].NIP) || (!naik && L[i].NIP >= R[j].NIP))) ||

(!berdasarkanNIP && ((naik && L[i].Nama <= R[j].Nama) || (!naik && L[i].Nama >= R[j].Nama)))) {

arr[k++] = L[i++];

}

else {

arr[k++] = R[j++];

}

perbandingan++;

pergeseran++;

}

while (i < n1) {

arr[k++] = L[i++];

pergeseran++;

}

while (j < n2) {

arr[k++] = R[j++];

pergeseran++;

}

}

int NIP(Pegawai arr[], int low, int high, bool naik, int& perbandingan) {

Pegawai pivot = arr[high];

int i = (low - 1);

for (int j = low; j <= high - 1; j++) {

if ((naik && arr[j].NIP < pivot.NIP) || (!naik && arr[j].NIP > pivot.NIP)) {

i++;

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

}

perbandingan++;

}

Tukar(arr[i + 1], arr[high]);

return (i + 1);

}

int Nama(Pegawai arr[], int low, int high, bool naik, int& perbandingan) {

Pegawai pivot = arr[high];

int i = (low - 1);

for (int j = low; j <= high - 1; j++) {

if ((naik && arr[j].Nama < pivot.Nama) || (!naik && arr[j].Nama > pivot.Nama)) {

i++;

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

}

perbandingan++;

}

Tukar(arr[i + 1], arr[high]);

return (i + 1);

}

int main() {

Pegawai pegawai[] = {

{101, "Dimas", "Jakarta", 'A'},

{103, "Anwar", "Bandung", 'B'},

{102, "Budi", "Surabaya", 'C'},

{104, "Jojo", "Yogyakarta", 'A'}

};

int n = sizeof(pegawai) / sizeof(pegawai[0]);

cout << "Data:\n";

tampilkanPegawai(pegawai, n);

int metode;

cout << "Pilih metode pengurutan:\n1. Quick Sort Rekursif\n2. Quick Sort Non-Rekursif\n3. Merge Sort\nInput: ";

cin >> metode;

int urutan;

cout << "Pilih urutan:\n1. Naik\n2. Turun\nInput: ";

cin >> urutan;

int berdasarkan;

cout << "Pilih pengurutan berdasarkan:\n1. NIP\n2. Nama\nInput: ";

cin >> berdasarkan;

int perbandingan = 0, pergeseran = 0;

switch (berdasarkan) {

case 1:

switch (metode) {

case 1:

quickSortRekursif(pegawai, 0, n - 1, urutan == 1, true, perbandingan, pergeseran);

break;

case 2:

quickSortNonRekursif(pegawai, 0, n - 1, urutan == 1, true, perbandingan, pergeseran);

break;

case 3:

merge(pegawai, 0, n - 1, urutan == 1, true, perbandingan, pergeseran);

break;

default:

cout << "Pilihan tidak valid.";

}

break;

case 2:

switch (metode) {

case 1:

quickSortRekursif(pegawai, 0, n - 1, urutan == 1, false, perbandingan, pergeseran);

break;

case 2:

quickSortNonRekursif(pegawai, 0, n - 1, urutan == 1, false, perbandingan, pergeseran);

break;

case 3:

merge(pegawai, 0, n - 1, urutan == 1, false, perbandingan, pergeseran);

break;

default:

cout << "Pilihan tidak valid.";

exit(0);

}

break;

default:

cout << "Pilihan tidak valid.";

exit(0);

}

cout << "Data setelah diurutkan:\n";

tampilkanPegawai(pegawai, n);

cout << "Total perbandingan: " << perbandingan << ", Total pergeseran: " << pergeseran << "\n";

}