Nama: jonathan gunawan

Kelas: TI-24-PA

Npm:242310020

TUGAS PERTEMUAN 2

1. Module halaman 24 nomor 1 dan 2 . Untuk nomor 2 buat dalam bentuk quick sort dan merge sort

   a. Nomor 1

   • QuickSort

quick sort nomor 1 buku[1].cpp

```cpp
using namespace std;

// Fungsi untuk menukar dua elemen dalam array
void swap (int arr[], int pos1, int pos2) {
    int temp; // Variabel sementara untuk penyimpanan
    temp = arr[pos1]; // Simpan nilai elemen di pos1
    arr[pos1] = arr [pos2]; // Ganti nilai elemen di pos1 dengan nilai elemen di pos2
    arr[pos2] = temp; // Ganti nilai elemen di pos2 dengan nilai yang disimpan di temp
}

// Fungsi untuk mempartisi array untuk quicksort
int partition (int arr[], int low, int high, int pivot) {
    int i = low; // Indeks untuk iterasi array
    int j = low; // Indeks untuk elemen yang lebih kecil dari pivot
    while (i <= high) { // Iterasi dari low hingga high
        if (arr[i] > pivot) { // Jika elemen saat ini lebih besar dari pivot
            i++; // Pindahkan indeks i ke elemen berikutnya
        }
        else { // Jika elemen saat ini kurang dari atau sama dengan pivot
            swap (arr, i, j); // Tukar elemen di i dan j
            i++; // Pindahkan indeks i ke elemen berikutnya
            j++; // Pindahkan indeks j ke elemen berikutnya (elemen yang lebih kecil dari pivot)
        }
    }
    return j-1; // Kembalikan posisi pivot setelah partisi
}

// Fungsi quicksort rekursif
void quicksort(int arr[], int low, int high) {
    if (low < high) { // Jika ada lebih dari satu elemen dalam sub-array
        int pivot = arr[high]; // Pilih elemen terakhir sebagai pivot
        int pos = partition(arr, low, high, pivot); // Partisi array dan dapatkan posisi pivot

        quicksort(arr, low, pos-1); // Urutkan sub-array kiri dari pivot
        quicksort(arr, pos+1, high); // Urutkan sub-array kanan dari pivot
    }
}

int main ()
{
    int n; // Variabel untuk menyimpan panjang array
    cout << "Tentukan panjang array = ";
    cin>> n; // Baca panjang array dari input

    int arr[n]; // Deklarasikan array dengan panjang n (perhatikan bahwa ini adalah VLA, yang tidak standar di C++)
```

```cpp
}

int main ()
{
    int n; // Variabel untuk menyimpan panjang array
    cout << "Tentukan panjang array = ";
    cin>> n; // Baca panjang array dari input

    int arr[n]; // Deklarasikan array dengan panjang n (perhatikan bahwa ini adalah VLA, yang tidak standar di C++)
    for (int i = 0; i < n; i++) { // Loop untuk membaca elemen-elemen array
        cin >> arr[i]; // Baca elemen array dari input
    }
    quicksort(arr, 0, n-1); // Panggil fungsi quicksort untuk mengurutkan array
    cout << "Berikutnya adalah array yang telah di sortir = ";
    for (int i = 0; i < n; i++) { // Loop untuk mencetak elemen-elemen array yang sudah diurutkan
        cout << arr[i] << "\t"; // Cetak elemen array diikuti dengan tab
    }
    cout << endl; // Cetak newline untuk baris baru.
    return 0; // Kembalikan 0 untuk menunjukkan eksekusi berhasil
}
```

- Mergesort

```cpp
#include <iostream>
using namespace std;

// Fungsi untuk menggabungkan dua subarray yang sudah terurut menjadi satu subarray terurut
void merge(int arr[], int l, int m, int r) {
    int x, y, z; // Indeks untuk iterasi array
    int n1 = m - l + 1; // Ukuran subarray kiri
    int n2 = r - m; // Ukuran subarray kanan

    int L[n1], R[n2]; // Array sementara untuk subarray kiri dan kanan

    // Salin data ke array sementara L[] dan R[]
    for (x = 0; x < n1; x++)
        L[x] = arr[l + x];
    for (y = 0; y < n2; y++)
        R[y] = arr[m + 1 + y];

    x = 0; // Indeks awal subarray pertama
    y = 0; // Indeks awal subarray kedua
    z = l; // Indeks awal subarray yang digabungkan

    // Gabungkan array sementara kembali ke arr[l..r]
    while (x < n1 && y < n2) {
        if (L[x] <= R[y]) { // Jika elemen L[x] lebih kecil atau sama dengan R[y]
            arr[z] = L[x]; // Salin L[x] ke arr[z]
            x++; // Pindahkan indeks subarray kiri
        } else { // Jika elemen R[y] lebih kecil dari L[x]
            arr[z] = R[y]; // Salin R[y] ke arr[z]
            y++; // Pindahkan indeks subarray kanan
        }
        z++; // Pindahkan indeks subarray yang digabungkan
    }

    // Salin elemen sisa dari L[], jika ada
    while (x < n1) {
        arr[z] = L[x];
        x++;
        z++;
    }

    // Salin elemen sisa dari R[], jika ada
    while (y < n2) {
        arr[z] = R[y];
        y++;
        z++;
    }
}

// Fungsi utama merge sort rekursif
void mergeSort(int arr[], int l, int r) {
    if (l < r) { // Jika ada lebih dari satu elemen
        int m = l + (r - l) / 2; // Temukan titik tengah

        // Urutkan setengah pertama dan setengah kedua
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);

        // Gabungkan setengah yang diurutkan
        merge(arr, l, m, r);
    }
}

// Fungsi untuk menampilkan array
void show(int A[], int size) {
    for (int i = 0; i < size; i++)
        cout << A[i] << " ";
}

int main() {
    int size; // Variabel untuk menyimpan ukuran array
    cout << "\nMasukan Banyak Data : ";
    cin >> size; // Baca ukuran array dari input

    int arr[size]; // Deklarasikan array dengan ukuran yang ditentukan (VLA warning)
    for (int i = 0; i < size; i++) {
        cout << "\nMasukan Data array ke-" << i << " : ";
        cin >> arr[i]; // Baca elemen array dari input
    }

    mergeSort(arr, 0, size - 1); // Panggil fungsi merge sort untuk mengurutkan array

    cout << "\nHasil\n";
    show(arr, size); // Tampilkan array yang sudah diurutkan

    return 0; // Kembalikan 0 untuk menunjukkan eksekusi berhasil
}
```

b. nomor 2

quicksort

```cpp
#include <iostream>
#include <vector>

using namespace std;

// Fungsi untuk menukar dua elemen
void swap(int &a, int &b) {
    int temp = a;
    a = b;
    b = temp;
}

// Fungsi untuk membagi array berdasarkan pivot
int partition(vector<int> &arr, int low, int high) {
    int pivot = arr[high]; // Pivot diambil dari elemen terakhir
    int i = low - 1;

    for (int j = low; j < high; j++) {
        if (arr[j] < pivot) { // Jika elemen lebih kecil dari pivot, tukar
            i++;
            swap(arr[i], arr[j]);
        }
    }
    swap(arr[i + 1], arr[high]); // Menempatkan pivot di posisi yang benar
    return (i + 1);
}

// Fungsi rekursif untuk melakukan Quick Sort
void quickSort(vector<int> &arr, int low, int high) {
    if (low < high) {
        int pivotIndex = partition(arr, low, high);

        // Rekursif untuk bagian kiri dan kanan dari pivot
        quickSort(arr, low, pivotIndex - 1);
```

```cpp
            cout << arr[i] << " ";
        }
        cout << endl;
    }

int main() {
    vector<int> arr;
    int n, element;

    // Output header dan meminta input dari user
    cout << "Program Quick Sort\n";
    cout << "Masukkan jumlah elemen: ";
    cin >> n;

    // Mengambil input array dari user
    cout << "Masukkan elemen array: ";
    for (int i = 0; i < n; i++) {
        cin >> element;
        arr.push_back(element);
    }

    // Menampilkan array sebelum diurutkan
    cout << "Array sebelum diurutkan: ";
    printArray(arr);

    // Melakukan Quick Sort
    quickSort(arr, 0, n - 1);

    // Menampilkan hasil array yang sudah diurutkan
    cout << "Array setelah diurutkan: ";
    printArray(arr);

    return 0;
```

mergesort

```cpp
#include <iostream>
using namespace std;

void merge(int arr[], int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;
    int leftArr[n1], rightArr[n2];

    for (int i = 0; i < n1; i++) leftArr[i] = arr[left + i];
    for (int i = 0; i < n2; i++) rightArr[i] = arr[mid + 1 + i];

    int i = 0, j = 0, k = left;
    while (i < n1 && j < n2) {
        if (leftArr[i] <= rightArr[j]) {
            arr[k++] = leftArr[i++];
        } else {
            arr[k++] = rightArr[j++];
        }
    }

    while (i < n1) arr[k++] = leftArr[i++];
    while (j < n2) arr[k++] = rightArr[j++];

    cout << "Menggabungkan List [";
    for (int i = left; i <= right; i++) {
        cout << arr[i] << (i < right ? ", " : "");
    }
    cout << "]\n";
}

void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;

        cout << "Pecah List [";
        for (int i = left; i <= right; i++) {
            cout << arr[i] << (i < right ? ", " : "");
        }
        cout << "]\n";

        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}
```

```cpp
int main() {
    int size;
    cout << "Masukkan jumlah elemen: ";
    cin >> size;

    int data[size];
    cout << "Masukkan " << size << " elemen: ";
    for (int i = 0; i < size; i++) {
        cin >> data[i];
    }

    cout << "Input data: [";
    for (int i = 0; i < size; i++) {
        cout << data[i] << (i < size - 1 ? ", " : "");
    }
    cout << "]\n";

    mergeSort(data, 0, size - 1);
    return 0;
}
```

Buatlah program quick sort dan merge sort dimana hasil output menunjukkan proses

Sorting

```cpp
#include <iostream>
using namespace std;

void printArray(int arr[], int size) {
    cout << "[";
    for (int i = 0; i < size; i++) {
        cout << arr[i] << (i < size - 1 ? " " : "");
    }
    cout << "]" << endl;
}

int partition(int arr[], int low, int high, int size) {
    int pivot = arr[high];
    int i = low - 1;

    for (int j = low; j < high; j++) {
        if (arr[j] > pivot) {
            i++;
            swap(arr[i], arr[j]);
        }
    }
    swap(arr[i + 1], arr[high]);

    printArray(arr, size);
    return (i + 1);
}

void quickSort(int arr[], int low, int high, int size) {
    if (low < high) {
        int pi = partition(arr, low, high, size);

        quickSort(arr, low, pi - 1, size);
        quickSort(arr, pi + 1, high, size);
    }
}

int main() {
    int numElements;
    cout << "Data yang akan di sort : ";
    cin >> numElements;

    int arr[numElements];
    cout << "Masukkan " << numElements << " angka: ";
    for (int i = 0; i < numElements; i++) {
        cin >> arr[i];
    }

    cout << "Quick Sort :" << endl;
    printArray(arr, numElements);

    quickSort(arr, 0, numElements - 1, numElements);
    return 0;
}
```