ALGORITMA DAN STUKTUR DATA

“Sorting Insertion Sort, Selection Sort”

Oleh :

Fina Salsabila Pramudita (5223600006)

Program Studi Sarjana Terapan Teknologi Game

Departemen Teknologi Multimedia Kreatif

Politeknik Elektronika Negeri Surabaya

2024

Percobaan

1. Buatlah workspace menggunakan Visual C++.

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#define MAX_SIZE 10
int Array[MAX_SIZE];

// Fungsi Straight Insertion Sort
void StraightInsertionSort() {
    int idx_i, idx_j, temp;
    for (idx_i = 1; idx_i < MAX_SIZE; idx_i++) {
        temp = Array[idx_i];
        idx_j = idx_i - 1;
        while (idx_j >= 0 && temp < Array[idx_j]) {
            Array[idx_j + 1] = Array[idx_j];
            idx_j--;
        }
        Array[idx_j + 1] = temp;
    }
}

int main() {
    int idx;
    srand((unsigned)time(0));
    //  Menggunakan  seed  berdasarkan  waktu  untuk  membangkitkan
bilangan acak

    // Membangkitkan bilangan acak
    std::cout << "DATA TIDAK URUT";
    for (idx = 0; idx < MAX_SIZE; idx++) {
        Array[idx] = rand() % 1000 + 1;
        // Memastikan bilangan acak antara 1 dan 1000
        std::cout << "\nData ke " << idx << " : " << Array[idx];
    }

    StraightInsertionSort();

    // Menampilkan data setelah terurut
```

```
        std::cout << "\nDATA TELAH URUT";
        for (idx = 0; idx < MAX_SIZE; idx++) {
            std::cout << "\nData ke " << idx << " : " << Array[idx];
        }

        return 0;
    }
```

2. Buatlah project untuk praktikum SORTING dan file C Source untuk metode pengurutan straight insertion sort, binary insertion sort dan selection sort.

```
    #include <iostream>
    #include <cstdlib>
    #include <ctime>
    #define ARRAY_SIZE 10

    int Data[ARRAY_SIZE];

    // Prototype Binary Insertion Sort
    void BinaryInsertionSort();

    int main() {
        srand(static_cast<unsigned int>(time(nullptr)));
        // Seed inisialisasi berdasarkan waktu sekarang
        std::cout << "DATA TIDAK URUT";

        // Mengisi array dengan bilangan acak
        for (int idx = 0; idx < ARRAY_SIZE; idx++) {
            Data[idx] = rand() % 1000 + 1;
            // Bilangan acak antara 1 dan 1000
            std::cout << "\nData ke " << idx << " : " << Data[idx];
        }

        BinaryInsertionSort(); // Pengurutan

        std::cout << "\nDATA SUDAH URUT";
        for (int idx = 0; idx < ARRAY_SIZE; idx++) {
            std::cout << "\nData ke " << idx << " : " << Data[idx];
        }
```

```
    return 0;
}

// Implementasi Binary Insertion Sort
void BinaryInsertionSort() {
    int i, j, left, right, mid, value;

    for (i = 1; i < ARRAY_SIZE; i++) {
        value = Data[i];
        left = 0;
        right = i - 1;

        while (left <= right) {
            mid = (left + right) / 2;
            if (value < Data[mid])
                right = mid - 1;
            else
                left = mid + 1;
        }

        for (j = i - 1; j >= left; j--)
            Data[j + 1] = Data[j];

        Data[left] = value;
    }
}
```

3. Cobalah untuk masing-masing percobaan di bawah dengan menambahkan menu pilihan metode pengurutan pada program utama.

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#define ARRAY_SIZE 10

int Data[ARRAY_SIZE];

// Menukar nilai antara dua variabel
void Swap(int *a, int *b) {
    int temp = *a;
```

```
        *a = *b;
        *b = temp;
    }

    // Pengurutan dengan metode Selection Sort
    void SelectionSorting() {
        int i, j, min_index;
        for (i = 0; i < ARRAY_SIZE - 1; i++) {
            min_index = i;
            for (j = i + 1; j < ARRAY_SIZE; j++) {
                if (Data[min_index] > Data[j]) {
                    min_index = j;
                }
            }
            Swap(&Data[i], &Data[min_index]);
        }
    }

    int main() {
        srand(static_cast<unsigned int>(time(nullptr)));
        // Inisialisasi pengacakan dengan seed waktu saat ini

        std::cout << "DATA TIDAK URUT";
        for (int idx = 0; idx < ARRAY_SIZE; idx++) {
            Data[idx] = rand() % 1000 + 1;  // Menghasilkan bilangan
    acak antara 1 dan 1000
            std::cout << "\nData ke " << idx << " : " << Data[idx];
        }

        SelectionSorting();  //  Melakukan  pengurutan  menggunakan
    Selection Sort

        std::cout << "\nDATA SUDAH URUT";
        for (int idx = 0; idx < ARRAY_SIZE; idx++) {
            std::cout << "\nData ke " << idx << " : " << Data[idx];
        }

        return 0;
}
```

Latihan

1. Tambahkan kode program untuk menampilkan perubahan setiap iterasi dari proses pengurutan dengan penyisipan langsung, penyisipan biner dan seleksi.

```
#include <iostream>
#include <cstdlib>
#include <ctime>

// Fungsi untuk menukar nilai antara dua variabel
void Swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

// Fungsi untuk mencetak isi array
void PrintArray(int *arr, int size) {
    for (int i = 0; i < size; i++) {
        std::cout << arr[i] << " ";
    }
    std::cout << std::endl;
}

// Fungsi pengurutan dengan metode Insertion Sort
void InsertionSort(int *arr, int size) {
    int i, j, key;
    for (i = 1; i < size; i++) {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}

// Fungsi pengurutan dengan metode Binary Insertion Sort
void BinaryInsertionSort(int *arr, int size) {
    int i, j, l, r, m, x;
```

```
    for (i = 1; i < size; i++) {
        x = arr[i];
        l = 0;
        r = i - 1;
        while (l <= r) {
            m = (l + r) / 2;
            if (x < arr[m])
                r = m - 1;
            else
                l = m + 1;
        }
        for (j = i - 1; j >= l; j--)
            arr[j + 1] = arr[j];
        arr[l] = x;
    }
}

// Fungsi pengurutan dengan metode Selection Sort
void SelectionSort(int *arr, int size) {
    int i, j, k;
    for (i = 0; i < size - 1; i++) {
        k = i;
        for (j = i + 1; j < size; j++) {
            if (arr[k] > arr[j]) {
                k = j;
            }
        }
        Swap(&arr[i], &arr[k]);
    }
}

int main() {
    srand(static_cast<unsigned int>(time(nullptr)));
    int choice, size;
    char repeat;

    do {
        std::cout << "Masukkan ukuran array: ";
        std::cin >> size;
```

```
        int *Data = new int[size];

        std::cout << "Data awal: ";
        for (int i = 0; i < size; i++) {
            Data[i] = rand() % 1000 + 1;
        }
        PrintArray(Data, size);
        std::cout << "\n";

        std::cout << "Pilih metode pengurutan:\n1. Insertion Sort\n2. Binary Insertion Sort\n3. Selection Sort\nPilihan Anda: ";
        std::cin >> choice;

        switch (choice) {
            case 1:
                InsertionSort(Data, size);
                break;
            case 2:
                BinaryInsertionSort(Data, size);
                break;
            case 3:
                SelectionSort(Data, size);
                break;
            default:
                std::cout << "Pilihan tidak valid!\n";
                break;
        }

        std::cout << "Data setelah diurutkan: ";
        PrintArray(Data, size);

        delete[] Data;

        std::cout << "\nUlangi program? (y/n): ";
        std::cin >> repeat;
        std::cout << "\n";
    } while (repeat == 'y' || repeat == 'Y');
```

```
    return 0;
}
```

2. Tambahkan kode program untuk menghitung banyaknya perbandingan dan pergeseran pada algoritma pengurutan penyisipan langsung, penyisipan biner dan seleksi.

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#define ARRAY_SIZE 10

int Array[ARRAY_SIZE];

void Swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

void PrintArray() {
    for (int idx = 0; idx < ARRAY_SIZE; idx++) {
        std::cout << Array[idx] << " ";
    }
    std::cout << std::endl;
}

void InsertionSort() {
    int i, j, key, comparisons = 0, shifts = 0;
    for (i = 1; i < ARRAY_SIZE; i++) {
        key = Array[i];
        j = i - 1;
        while (j >= 0 && Array[j] > key) {
            comparisons++;
            Array[j + 1] = Array[j];
            shifts++;
            j--;
        }
        Array[j + 1] = key;
        if (j >= 0) comparisons++;
    }
```

```
    std::cout << "Total perbandingan: " << comparisons << ", Total
pergeseran: " << shifts << std::endl;
}

void BinaryInsertionSort() {
    int i, j, l, r, m, x, comparisons = 0, shifts = 0;
    for (i = 1; i < ARRAY_SIZE; i++) {
        x = Array[i];
        l = 0;
        r = i - 1;
        while (l <= r) {
            m = (l + r) / 2;
            comparisons++;
            if (x < Array[m])
                r = m - 1;
            else
                l = m + 1;
        }
        for (j = i - 1; j >= l; j--) {
            Array[j + 1] = Array[j];
            shifts++;
        }
        Array[l] = x;
    }
    std::cout << "Total perbandingan: " << comparisons << ", Total
pergeseran: " << shifts << std::endl;
}

void SelectionSort() {
    int i, j, k, comparisons = 0, swaps = 0;
    for (i = 0; i < ARRAY_SIZE - 1; i++) {
        k = i;
        for (j = i + 1; j < ARRAY_SIZE; j++) {
            comparisons++;
            if (Array[k] > Array[j]) {
                k = j;
            }
        }
        if (k != i) {
```

```
            Swap(&Array[i], &Array[k]);
            swaps++;
        }
    }
    std::cout << "Total perbandingan: " << comparisons << ", Total
tukar: " << swaps << std::endl;
}

int main() {
    srand(static_cast<unsigned int>(time(nullptr)));
    int choice;
    char repeat;

    do {
        std::cout << "Data awal: ";
        for (int idx = 0; idx < ARRAY_SIZE; idx++) {
            Array[idx] = rand() % 1000 + 1;
            std::cout << Array[idx] << " ";
        }
        std::cout << "\n\n";

        std::cout << "Pilih metode pengurutan:\n1. Insertion
Sort\n2. Binary Insertion Sort\n3. Selection Sort\nPilihan Anda:
";
        std::cin >> choice;

        switch (choice) {
            case 1:
                InsertionSort();
                break;
            case 2:
                BinaryInsertionSort();
                break;
            case 3:
                SelectionSort();
                break;
            default:
                std::cout << "Pilihan tidak valid!\n";
                break;
```

```
        }

        std::cout << "Ulangi program? (y/n): ";
        std::cin >> repeat;
        std::cout << "\n";
    } while (repeat == 'y' || repeat == 'Y');

    return 0;
}
```

3. Buatlah project baru untuk Latihan dan implementasikan pengurutan data Pegawai pada tugas pendahuluan dengan ketentuan :.

   a. Metode pengurutan dapat dipilih.

   b. Pengurutan dapat dipilih secara urut naik atau turun.

   c. Pengurutan dapat dipilih berdasarkan NIP dan NAMA.

   d. Gunakan struktur data array.

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <ctime>
#include <cstdlib>

// Definisi struktur baru untuk Karyawan
struct Karyawan {
std::string id;
std::string nama;
};

// Fungsi untuk mencetak informasi Karyawan
void printKaryawan(const std::vector<Karyawan>& karyawan) {
for (const auto& k : karyawan) {
std::cout << "ID: " << k.id << ", Nama: " << k.nama << std::endl;
}
}

// Fungsi pengurutan dengan metode Selection Sort
```

```
void selectionSort(std::vector<Karyawan>& karyawan, bool sortById,
bool ascending) {
int n = karyawan.size();
for (int i = 0; i < n - 1; i++) {
int idx = i;
for (int j = i + 1; j < n; j++) {
if (sortById) {
if ((ascending && karyawan[j].id < karyawan[idx].id) || (!ascending
&& karyawan[j].id > karyawan[idx].id)) {
idx = j;
}
} else {
if ((ascending && karyawan[j].nama < karyawan[idx].nama) ||
(!ascending && karyawan[j].nama > karyawan[idx].nama)) {
idx = j;
}
}
}
if (idx != i) {
std::swap(karyawan[idx], karyawan[i]);
}
}
}

// Fungsi untuk mengacak nama pegawai
void acakNama(std::vector<Karyawan>& karyawan) {
std::vector<std::string> nama = {"Erik", "Kevin", "John",
"Giselle"};
std::srand(static_cast<unsigned int>(std::time(nullptr)));
for (auto& k : karyawan) {
int idx = std::rand() % nama.size();
k.nama = nama[idx];
}
}

int main() {
std::vector<Karyawan> karyawan = {
{"123456789", "Erik"},
{"234567890", "Kevin"},
```

```
    {"345678901", "John"},
    {"456789012", "Giselle"}
    };

    // Mengacak urutan karyawan dan nama mereka
    std::srand(static_cast<unsigned int>(std::time(nullptr)));
    std::random_shuffle(karyawan.begin(), karyawan.end());
    acakNama(karyawan);

    char repeat;
    do {
    int choice, order;
    std::cout << "Pilih metode pengurutan:\n1. Berdasarkan ID\n2.
    Berdasarkan Nama\nPilihan Anda: ";
    std::cin >> choice;
    std::cout << "Pilih urutan:\n1. Naik\n2. Turun\nPilihan Anda: ";
    std::cin >> order;

    bool sortById = (choice == 1);
    bool ascending = (order == 1);

    selectionSort(karyawan, sortById, ascending);

    std::cout << "Data karyawan setelah diurutkan:" << std::endl;
    printKaryawan(karyawan);

    std::cout << "Apakah Anda ingin mengurutkan lagi? (y/n): ";
    std::cin >> repeat;
    } while (repeat == 'y' || repeat == 'Y');

    return 0;
    }
```

4. Berikan kesimpulan dari percobaan dan latihan yang telah Anda lakukan.
   - Kecepatan dan Efisiensi

     Insertion Sort: Lebih efisien untuk data yang sudah hampir terurut atau memiliki jumlah elemen yang sedikit.

Selection Sort: Lebih lambat daripada Insertion Sort dalam sebagian besar kasus, terutama untuk jumlah data yang besar, karena memiliki kompleksitas waktu yang lebih tinggi.

- Kompleksitas Waktu

  Insertion Sort: Memiliki kompleksitas waktu rata-rata O(n^2), di mana n adalah jumlah elemen dalam array.

  Selection Sort: Memiliki kompleksitas waktu rata-rata O(n^2), sama dengan Insertion Sort.

- Konsep Algoritma

  Insertion Sort: Menggunakan konsep membandingkan elemen-elemen satu per satu dan menyisipkan elemen baru ke posisi yang sesuai dalam bagian yang sudah terurut.

  Selection Sort: Menggunakan konsep memilih elemen terkecil (atau terbesar) dari bagian yang belum terurut dan menukar dengan elemen pertama (atau terakhir) dalam bagian yang belum terurut.