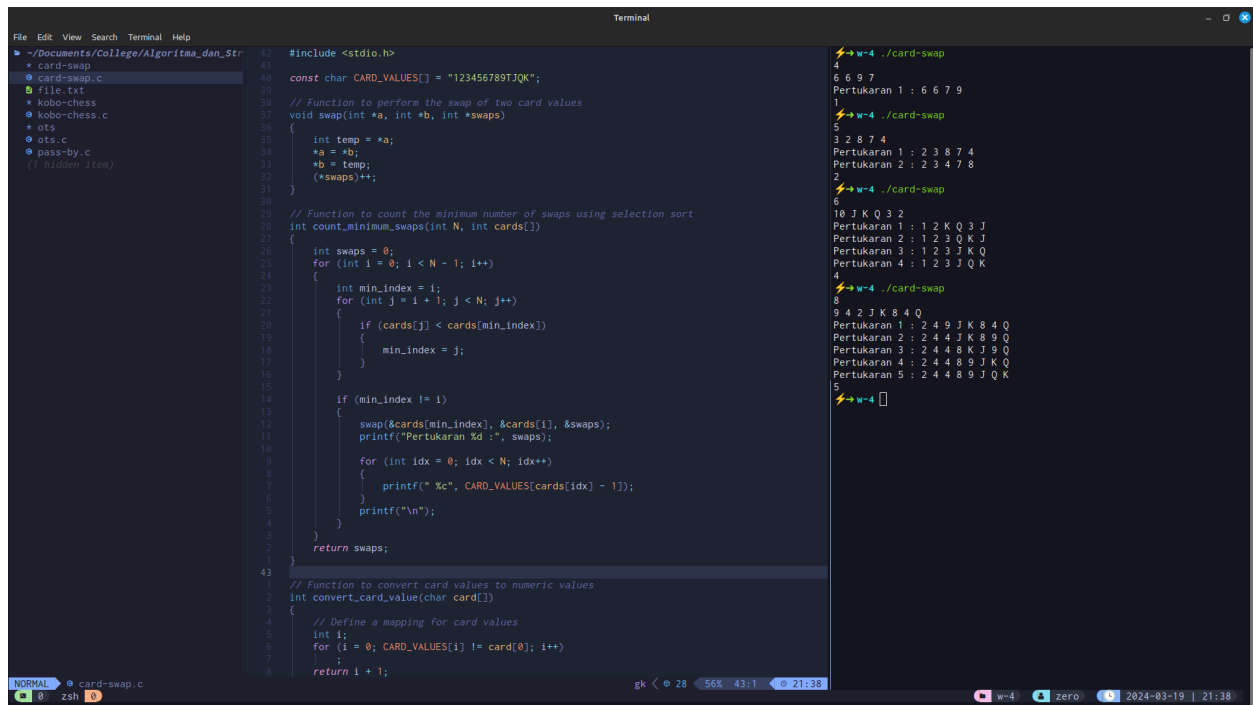


Nama : Arif Widiyanto
NIM : 1203230051
Tugas : OTH Week 4
Mata Kuliah : Algoritma & Struktur Data

1. Mengurutkan Kartu Refan

1.1. Penulisan Kode beserta output



```
#include <stdio.h>

const char CARD_VALUES[] = "123456789TJQK";

// Function to perform the swap of two card values
void swap(int *a, int *b, int *swaps)
{
    int temp = *a;
    *a = *b;
    *b = temp;
    (*swaps)++;
}

// Function to count the minimum number of swaps using selection sort
int count_minimum_swaps(int N, int cards[])
{
    int swaps = 0;
    for (int i = 0; i < N - 1; i++)
    {
        int min_index = i;
        for (int j = i + 1; j < N; j++)
        {
            if (cards[j] < cards[min_index])
            {
                min_index = j;
            }
        }
        if (min_index != i)
        {
            swap(&cards[min_index], &cards[i], &swaps);
            printf("Pertukaran %d : ", swaps);
            for (int idx = 0; idx < N; idx++)
            {
                printf(" %c", CARD_VALUES[cards[idx] - 1]);
            }
            printf("\n");
        }
    }
    return swaps;
}

// Function to convert card values to numeric values
int convert_card_value(char card[])
{
    // Define a mapping for card values
    int i;
    for (i = 0; CARD_VALUES[i] != card[0]; i++)
    {
    }
    return i + 1;
}

int main()
{
    int cards[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
    int N = sizeof(cards) / sizeof(int);
    int swaps = count_minimum_swaps(N, cards);
    printf("Total swaps: %d\n", swaps);
    for (int i = 0; i < N; i++)
    {
        printf(" %c", CARD_VALUES[cards[i] - 1]);
    }
    printf("\n");
    return 0;
}
```

```
#include <stdio.h>

const char CARD_VALUES[] = "123456789TJQK";

// Function to perform the swap of two card values
void swap(int *a, int *b, int *swaps)
{
    int temp = *a;
    *a = *b;
    *b = temp;
    (*swaps)++;
}

// Function to count the minimum number of swaps using selection sort
int count_minimum_swaps(int N, int cards[])
{
    int swaps = 0;
```

```

for (int i = 0; i < N - 1; i++)
{
    int min_index = i;
    for (int j = i + 1; j < N; j++)
    {
        if (cards[j] < cards[min_index])
        {
            min_index = j;
        }
    }

    if (min_index != i)
    {
        swap(&cards[min_index], &cards[i], &swaps);

        printf("Pertukaran %d :", swaps);
        for (int idx = 0; idx < N; idx++)
        {
            printf(" %c", CARD_VALUES[cards[idx] - 1]);
        }
        printf("\n");
    }
}
return swaps;
}

// Function to convert card values to numeric values
int convert_card_value(char card[])
{
    // Define a mapping for card values
    int i;
    for (i = 0; CARD_VALUES[i] != card[0]; i++)
        ;
    return i + 1;
}

int main()
{
    int N;
    scanf("%d", &N); // Read the number of cards
    int cards[N];

```

```

for (int i = 0; i < N; i++)
{
    char card[3];
    scanf("%s", card);

    // Convert card values to numeric values
    cards[i] = convert_card_value(card);
}

// Call the function to count the minimum number of swaps
int minimal_swaps = count_minimum_swaps(N, cards);

// Output the minimum number of swaps
printf("%d\n", minimal_swaps);

return 0;
}

```

1.2. Penjelasan Kode

1. `#include <stdio.h>`

Baris ini mengimpor file header standar input/output, yang menyediakan fungsi seperti printf dan scanf.

2. `const char CARD_VALUES[] = "123456789TJQK";`

Baris ini mendefinisikan sebuah array karakter konstan CARD_VALUES yang berisi nilai-nilai mungkin dari kartu. Ini mewakili nilai numerik dan karakter untuk kartu dari 1 hingga K.

3.

```

void swap(int *a, int *b, int *swaps)
{
    int temp = *a;
    *a = *b;
    *b = temp;
    (*swaps)++;
}

```

Fungsi `swap` ini mengambil tiga *parameter*: *pointer* ke *integer* a dan b yang mewakili nilai yang akan ditukar, dan *pointer* ke *integer* swaps yang mewakili

jumlah pertukaran. Ini menukar nilai a dan b, kemudian meningkatkan jumlah pertukaran.

4.

```
int count_minimum_swaps(int N, int cards[])
{
    int swaps = 0;
    for (int i = 0; i < N - 1; i++)
    {
        int min_index = i;
        for (int j = i + 1; j < N; j++)
        {
            if (cards[j] < cards[min_index])
            {
                min_index = j;
            }
        }

        if (min_index != i)
        {
            swap(&cards[min_index], &cards[i], &swaps);
            printf("Pertukaran %d :", swaps);

            for (int idx = 0; idx < N; idx++)
            {
                printf(" %c", CARD_VALUES[cards[idx] - 1]);
            }
            printf("\n");
        }
    }
    return swaps;
}
```

Fungsi `count_minimum_swaps` menghitung jumlah minimum pertukaran menggunakan algoritma *selection sort*. Ini memerlukan dua parameter: *integer* `N` yang merupakan jumlah kartu, dan *array integer* `cards` yang berisi nilai-nilai kartu. Fungsi ini menggunakan dua *loop* bersarang untuk membandingkan nilai-nilai kartu dan menukar mereka jika diperlukan. Jumlah pertukaran dilacak dalam variabel `swaps`.

5.

```
int convert_card_value(char card[])
{
    int i;
    for (i = 0; CARD_VALUES[i] != card[0]; i++)
        ;
    return i + 1;
}
```

Fungsi `convert_card_value` mengonversi nilai-nilai kartu menjadi nilai numerik. Ini mengambil *string* `card` yang mewakili nilai kartu dan mencari nilai numerik yang sesuai dalam array `CARD_VALUES`. Kemudian, nilai tersebut dikembalikan.

6.

```
int main()
{
    int N;
    scanf("%d", &N); // Membaca jumlah kartu

    int cards[N];
    for (int i = 0; i < N; i++)
    {
        char card[3];
        scanf("%s", card);

        // Mengonversi nilai kartu ke nilai numerik
        cards[i] = convert_card_value(card);
    }

    // Memanggil fungsi untuk menghitung jumlah minimum pertukaran
    int minimal_pertukaran = count_minimum_swaps(N, cards);

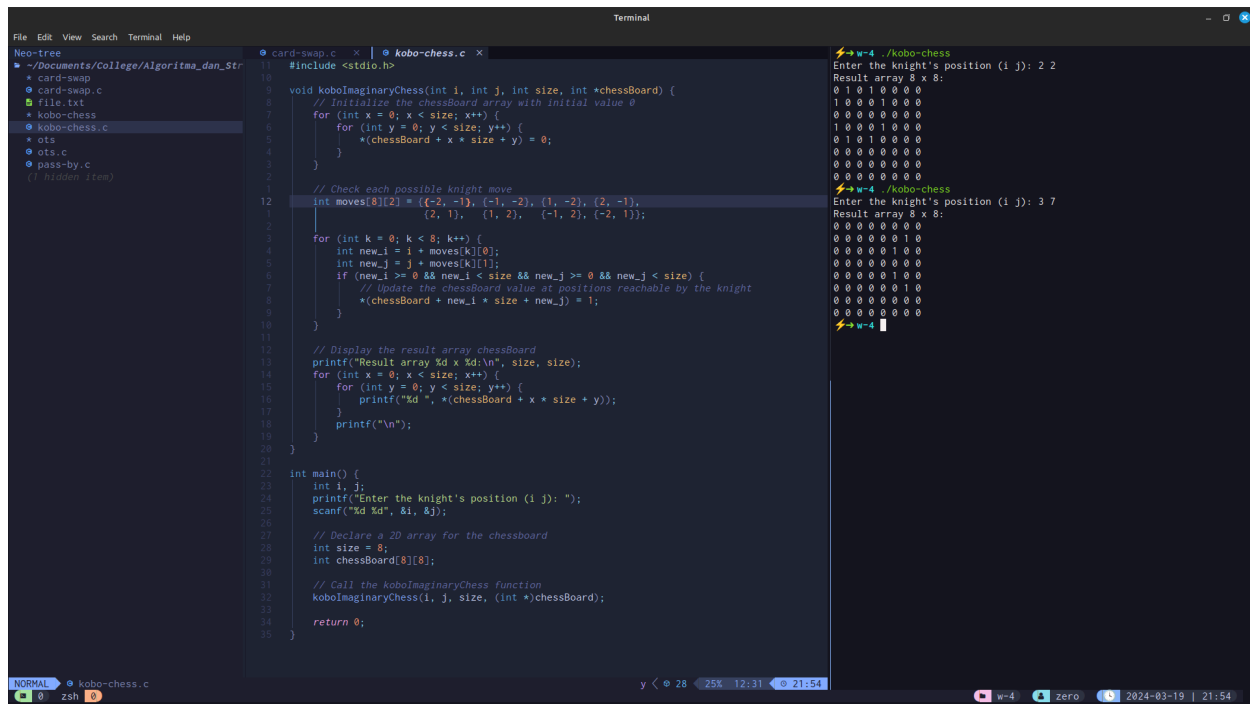
    // Output jumlah minimum pertukaran
    printf("%d\n", minimal_pertukaran);

    return 0;
}
```

Fungsi `main` ini adalah tempat dimulainya eksekusi program. Pertama, ia membaca jumlah kartu yang akan dimainkan. Kemudian, ia membaca nilai-nilai kartu dan mengonversinya menjadi nilai numerik. Selanjutnya, ia memanggil fungsi `count_minimum_swaps` untuk menghitung jumlah minimum pertukaran yang diperlukan untuk mengurutkan kartu. Hasilnya dicetak ke layar.

2. Kobo Imaginary Chess

2.1. Penulisan Kode beserta output



```
File Edit View Search Terminal Help
Neo-tree
* ~/Documents/College/Algoritma_dan_Str
  * card-swap
  * file.txt
  * kobo-chess
  * kobo-chess.c
  * ots
  * ots.c
  * pass-by.c
  * (hidden files)

card-swap.c x kobo-chess.c x
#include <stdio.h>

void koboImaginaryChess(int i, int j, int size, int *chessBoard) {
    // Initialize the chessBoard array with initial value 0
    for (int x = 0; x < size; x++) {
        for (int y = 0; y < size; y++) {
            *(chessBoard + x * size + y) = 0;
        }
    }

    // Check each possible knight move
    int moves[8][2] = {{-2, -1}, {-1, -2}, {1, -2}, {2, -1},
                      {2, 1}, {1, 2}, {-1, 2}, {-2, 1}};

    for (int k = 0; k < 8; k++) {
        int new_i = i + moves[k][0];
        int new_j = j + moves[k][1];
        if (new_i >= 0 && new_i < size && new_j >= 0 && new_j < size) {
            // Update the chessBoard value at positions reachable by the knight
            *(chessBoard + new_i * size + new_j) = 1;
        }
    }

    // Display the result array chessBoard
    printf("Result array %d x %d:\n", size, size);
    for (int x = 0; x < size; x++) {
        for (int y = 0; y < size; y++) {
            printf("%d ", *(chessBoard + x * size + y));
        }
        printf("\n");
    }
}

int main() {
    int i, j;
    printf("Enter the knight's position (i j): ");
    scanf("%d %d", &i, &j);

    // Declare a 2D array for the chessboard
    int size = 8;
    int chessBoard[8][8];

    // Call the koboImaginaryChess function
    koboImaginaryChess(i, j, size, (int *)chessBoard);

    return 0;
}

NORMAL kobo-chess.c
y < 28 25% 12:31 21:54

w-4 /kobo-chess
Enter the knight's position (i j): 2 2
Result array 8 x 8:
0 1 0 1 0 0 0 0
1 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0
1 0 0 0 1 0 0 0
0 1 0 1 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

w-4 /kobo-chess
Enter the knight's position (i j): 3 7
Result array 8 x 8:
0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

w-4
```

```
#include <stdio.h>

void koboImaginaryChess(int i, int j, int size, int *chessBoard) {
    // Initialize the chessBoard array with initial value 0
    for (int x = 0; x < size; x++) {
        for (int y = 0; y < size; y++) {
            *(chessBoard + x * size + y) = 0;
        }
    }

    // Check each possible knight move
    int moves[8][2] = {{-2, -1}, {-1, -2}, {1, -2}, {2, -1},
```

```

        {2, 1},    {1, 2},    {-1, 2}, {-2, 1}};

for (int k = 0; k < 8; k++) {
    int new_i = i + moves[k][0];
    int new_j = j + moves[k][1];
    if (new_i >= 0 && new_i < size && new_j >= 0 && new_j < size) {
        // Update the chessBoard value at positions reachable by the
knight
        *(chessBoard + new_i * size + new_j) = 1;
    }
}

// Display the result array chessBoard
printf("Result array %d x %d:\n", size, size);
for (int x = 0; x < size; x++) {
    for (int y = 0; y < size; y++) {
        printf("%d ", *(chessBoard + x * size + y));
    }
    printf("\n");
}
}

int main() {
    int i, j;
    printf("Enter the knight's position (i j): ");
    scanf("%d %d", &i, &j);

    // Declare a 2D array for the chessboard
    int size = 8;
    int chessBoard[8][8];

    // Call the koboImaginaryChess function
    koboImaginaryChess(i, j, size, (int *)chessBoard);

    return 0;
}

```

2.2. Penjelasan Kode

1. `#include <stdio.h>`

Baris ini mengimpor file header standar input/output, yang menyediakan fungsi seperti printf dan scanf.

2.

```
void koboImaginaryChess(int i, int j, int size, int *chessBoard) {
    // Initialize the chessBoard array with initial value 0
    for (int x = 0; x < size; x++) {
        for (int y = 0; y < size; y++) {
            *(chessBoard + x * size + y) = 0;
        }
    }

    // Check each possible knight move
    int moves[8][2] = {{-2, -1}, {-1, -2}, {1, -2}, {2, -1},
                       {2, 1}, {1, 2}, {-1, 2}, {-2, 1}};

    for (int k = 0; k < 8; k++) {
        int new_i = i + moves[k][0];
        int new_j = j + moves[k][1];
        if (new_i >= 0 && new_i < size && new_j >= 0 && new_j < size) {
            // Update the chessBoard value at positions reachable by the
            knight
            *(chessBoard + new_i * size + new_j) = 1;
        }
    }

    // Display the result array chessBoard
    printf("Result array %d x %d:\n", size, size);
    for (int x = 0; x < size; x++) {
        for (int y = 0; y < size; y++) {
            printf("%d ", *(chessBoard + x * size + y));
        }
        printf("\n");
    }
}
```

Fungsi ini bertujuan untuk mensimulasikan gerakan kuda pada papan catur. Pertama-tama, fungsi ini menginisialisasi papan catur dengan ukuran yang telah ditentukan (`size`) dan mengatur semua kotak menjadi kosong (nilai 0). Selanjutnya, fungsi ini memeriksa setiap kemungkinan gerakan kuda dengan menggunakan array `moves`, di mana setiap gerakan direpresentasikan sebagai

perubahan koordinat. Jika posisi baru yang dihasilkan masih berada dalam batas papan catur, maka nilai yang sesuai pada papan catur diubah menjadi 1, menandakan bahwa kotak tersebut dapat dijangkau oleh kuda dalam satu langkah. Akhirnya, fungsi ini mencetak papan catur hasilnya ke layar untuk ditampilkan kepada pengguna.

3.

```
int main() {
    int i, j;
    printf("Enter the knight's position (i j): ");
    scanf("%d %d", &i, &j);

    // Declare a 2D array for the chessboard
    int size = 8;
    int chessBoard[8][8];

    // Call the koboImaginaryChess function
    koboImaginaryChess(i, j, size, (int *)chessBoard);

    return 0;
}
```

Fungsi `main` adalah titik masuk utama program. Pertama-tama, pengguna diminta untuk memasukkan posisi awal kuda (koordinat i dan j). Kemudian, array `chessBoard` dengan ukuran 8x8 dideklarasikan untuk merepresentasikan papan catur. Fungsi `koboImaginaryChess` dipanggil dengan posisi awal kuda yang dimasukkan pengguna, ukuran papan catur, dan array `chessBoard`. Setelah itu, program selesai dieksekusi.

Kedua fungsi ini bekerja bersama untuk mensimulasikan gerakan kuda pada papan catur dan menampilkan papan catur hasilnya kepada pengguna. Fungsi `koboImaginaryChess` bertanggung jawab untuk perhitungan gerakan kuda, sementara `main` bertanggung jawab untuk mengatur aliran program dan interaksi dengan pengguna.