

NAMA : FELIX SIMAMORA

NIM : 1203230068

MATKUL: PRAKTIKUM ALGORITMA DAN STRUKTUR DATA

NO.1

```
#include <stdio.h>
#include <stdlib.h>

// Fungsi untuk menghitung jumlah pertukaran minimum
int hitung_pertukaran(int arr[], int n) {
    int jumlah_pertukaran = 0;
    // Lakukan bubble sort untuk mengurutkan array
    for (int i = 0; i < n-1; i++) {
        for (int j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                // Tukar posisi elemen
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
                jumlah_pertukaran++;
            }
        }
    }
    return jumlah_pertukaran;
}

int main() {
    int n;
    printf("Masukkan jumlah kartu: ");
    scanf("%d", &n);

    int *kartu = (int *)malloc(n * sizeof(int));

    printf("Masukkan nilai kartu: ");
    for (int i = 0; i < n; i++) {
        char nilai_kartu[3];
        scanf("%s", nilai_kartu);
        // Mengubah nilai kartu menjadi bilangan bulat
        if (nilai_kartu[0] >= '2' && nilai_kartu[0] <= '9') {
            kartu[i] = nilai_kartu[0] - '0';
        } else {
            switch (nilai_kartu[0]) {
```

```

        case '1':
            kartu[i] = 10;
            break;
        case 'J':
            kartu[i] = 11;
            break;
        case 'Q':
            kartu[i] = 12;
            break;
        case 'K':
            kartu[i] = 13;
            break;
    }
}

int jumlah_pertukaran = hitung_pertukaran(kartu, n);
printf("Jumlah minimal langkah pertukaran: %d\n", jumlah_pertukaran);

free(kartu);
return 0;
}

```

The screenshot shows the Visual Studio Code interface with a C program open in the editor. The program is a swap sort algorithm that takes an array of cards and returns the minimum number of swaps required to sort them. The terminal shows the program being compiled and executed with sample input.

```

C bbl.c > main()
22 int main() {
42     kartu[i] = 11;
43     break;
44     case 'Q':
45     kartu[i] = 12;
46     break;
47     case 'K':
48     kartu[i] = 13;
49     break;
50 }
51 }
52
53
54 int jumlah_pertukaran = hitung_pertukaran(kartu, n);
55 printf("Jumlah minimal langkah pertukaran: %d\n", jumlah_pertukaran);
56
57 free(kartu);
58 return 0;
59 }
60
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\aww\alpro> cd "d:\aww\alpro\" ; if ($?) { gcc bbl.c -o bbl }
Masukkan jumlah kartu: 4
Masukkan nilai kartu: 6 6 9 7
Jumlah minimal langkah pertukaran: 1
PS D:\aww\alpro>

```

```

#include <stdio.h>
#include <stdlib.h>

// Fungsi untuk menghitung jumlah pertukaran minimum
int hitung_pertukaran(int arr[], int n) {
    int jumlah_pertukaran = 0;
    // Lakukan bubble sort untuk mengurutkan array
    for (int i = 0; i < n-1; i++) {
        for (int j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                // Tukar posisi elemen
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
                jumlah_pertukaran++;

                // Tampilkan urutan kartu setelah pertukaran
                printf("Urutan kartu setelah pertukaran ke-%d: ",
jumlah_pertukaran);
                for (int k = 0; k < n; k++) {
                    printf("%d ", arr[k]);
                }
                printf("\n");
            }
        }
    }
    return jumlah_pertukaran;
}

int main() {
    int n;
    printf("Masukkan jumlah kartu: ");
    scanf("%d", &n);

    int *kartu = (int *)malloc(n * sizeof(int));

    printf("Masukkan nilai kartu: ");
    for (int i = 0; i < n; i++) {
        char nilai_kartu[3];
        scanf("%s", nilai_kartu);
        // Mengubah nilai kartu menjadi bilangan bulat
        if (nilai_kartu[0] >= '2' && nilai_kartu[0] <= '9') {
            kartu[i] = nilai_kartu[0] - '0';
        } else {
            switch (nilai_kartu[0]) {

```

```

        case '1':
            kartu[i] = 10;
            break;
        case 'J':
            kartu[i] = 11;
            break;
        case 'Q':
            kartu[i] = 12;
            break;
        case 'K':
            kartu[i] = 13;
            break;
    }
}

int jumlah_pertukaran = hitung_pertukaran(kartu, n);
printf("Jumlah minimal langkah pertukaran: %d\n", jumlah_pertukaran);

free(kartu);
return 0;
}

```

The screenshot shows the Visual Studio Code interface with the following components:

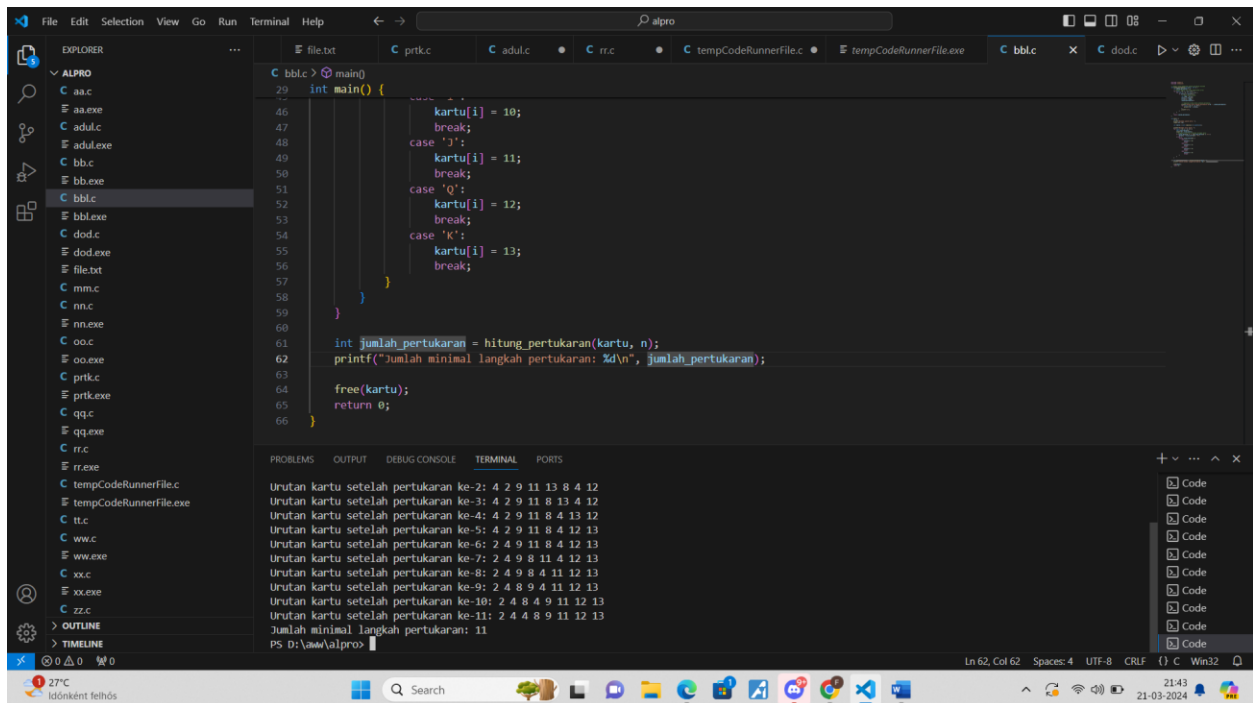
- EXPLORER:** A file explorer on the left showing a project named 'ALPRO' with various files like 'aa.c', 'adul.c', 'bb.c', etc.
- EDITOR:** The main window showing the C code from the previous block, with line numbers 29 to 66.
- TERMINAL:** The bottom panel showing the execution of the program. It starts with a command prompt where the user enters the number of cards (8) and the card values (9 4 2 7 8 4 Q). The program then displays the sequence of cards after each of the 9 swaps.
- PROBLEMS:** A panel on the right showing no errors or warnings.

Terminal Output:

```

PS D:\aaa\alpro> cd "d:\aaa\alpro" ; if ($?) { gcc bbl.c -o bbl } ; if ($?) { .\bbl }
Masukkan jumlah kartu: 8
Masukkan nilai kartu: 9 4 2 7 8 4 Q
Urutan kartu setelah pertukaran ke-1: 4 9 2 11 13 8 4 12
Urutan kartu setelah pertukaran ke-2: 4 2 9 11 13 8 4 12
Urutan kartu setelah pertukaran ke-3: 4 2 9 11 8 13 4 12
Urutan kartu setelah pertukaran ke-4: 4 2 9 11 8 4 13 12
Urutan kartu setelah pertukaran ke-5: 4 2 9 11 8 4 12 13
Urutan kartu setelah pertukaran ke-6: 2 4 9 11 8 4 12 13
Urutan kartu setelah pertukaran ke-7: 2 4 9 8 11 4 12 13
Urutan kartu setelah pertukaran ke-8: 2 4 9 8 4 11 12 13
Urutan kartu setelah pertukaran ke-9: 2 4 8 9 4 11 12 13

```



NO.2

```
#include <stdio.h>
#include <stdlib.h>

void koboImaginaryChess(int i, int j, int size, int *chessBoard) {
    // Mengecek apakah posisi i dan j berada dalam rentang valid
    if (i < 0 || i >= size || j < 0 || j >= size) {
        printf("Posisi tidak valid.\n");
        return;
    }

    // Array offset yang merepresentasikan langkah-langkah yang mungkin oleh knight
    int offset[8][2] = {{-2, -1}, {-2, 1}, {-1, -2}, {-1, 2}, {1, -2}, {1, 2}, {2, -1}, {2, 1}};

    // Melakukan iterasi untuk setiap kemungkinan langkah knight
    for (int k = 0; k < 8; k++) {
        int x = i + offset[k][0];
        int y = j + offset[k][1];

        // Mengecek apakah langkah masih berada dalam papan catur
        if (x >= 0 && x < size && y >= 0 && y < size) {
            // Menandai posisi yang dapat dicapai oleh knight dengan nilai 1
        }
    }
}
```

```

        *(chessBoard + x * size + y) = 1;
    }
}

int main() {
    int i, j;
    int size = 8; // Ukuran papan catur

    // Membuat array 2D untuk papan catur
    int *chessBoard = (int *)malloc(size * size * sizeof(int));
    if (chessBoard == NULL) {
        printf("Alokasi memori gagal.\n");
        return 1;
    }

    // Inisialisasi papan catur dengan nilai 0
    for (int x = 0; x < size; x++) {
        for (int y = 0; y < size; y++) {
            *(chessBoard + x * size + y) = 0;
        }
    }

    // Membaca input posisi bidak kuda
    printf("Masukkan nilai i dan j (0 <= i, j < 8): ");
    scanf("%d %d", &i, &j);

    // Memanggil fungsi untuk menentukan posisi yang dapat dicapai oleh bidak
    kuda
    koboImaginaryChess(i, j, size, chessBoard);

    // Menampilkan papan catur dengan posisi yang dapat dicapai oleh bidak kuda
    printf("Papan Catur dengan Posisi yang Dapat Dicapai oleh Bidak Kuda:\n");
    for (int x = 0; x < size; x++) {
        for (int y = 0; y < size; y++) {
            printf("%d ", *(chessBoard + x * size + y));
        }
        printf("\n");
    }

    // Membebaskan memori yang dialokasikan untuk papan catur
    free(chessBoard);

    return 0;
}

```

```

27 int main() {
28     // Masukkan nilai i dan j (0 <= i, j < 8):
29     printf("Masukkan nilai i dan j (0 <= i, j < 8): ");
30     scanf("%d %d", &i, &j);
31
32     // Mengambil fungsi untuk menentukan posisi yang dapat dicapai oleh bidak kuda
33     koboImaginaryChess(i, j, size, chessBoard);
34
35     // Menampilkan papan catur dengan posisi yang dapat dicapai oleh bidak kuda
36     printf("Papan Catur dengan Posisi yang Dapat Dicapai oleh Bidak Kuda:\n");
37     for (int x = 0; x < size; x++) {
38         for (int y = 0; y < size; y++) {
39             printf("%d ", (chessBoard + x * size + y));
40         }
41         printf("\n");
42     }
43
44     // Membebaskan memori yang dialokasikan untuk papan catur
45     free(chessBoard);
46
47     return 0;
48 }

```

Terminal Output:

```

PS D:\alpro> cd "d:\alpro"; if ($?) { gcc dod.c -o dod }; if ($?) { .\dod }
Masukkan nilai i dan j (0 <= i, j < 8): 2 2
Papan Catur dengan Posisi yang Dapat Dicapai oleh Bidak Kuda:
0 1 0 1 0 0 0 0
1 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0
1 0 0 1 0 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

```

```

27 int main() {
28     // Masukkan nilai i dan j (0 <= i, j < 8):
29     printf("Masukkan nilai i dan j (0 <= i, j < 8): ");
30     scanf("%d %d", &i, &j);
31
32     // Mengambil fungsi untuk menentukan posisi yang dapat dicapai oleh bidak kuda
33     koboImaginaryChess(i, j, size, chessBoard);
34
35     // Menampilkan papan catur dengan posisi yang dapat dicapai oleh bidak kuda
36     printf("Papan Catur dengan Posisi yang Dapat Dicapai oleh Bidak Kuda:\n");
37     for (int x = 0; x < size; x++) {
38         for (int y = 0; y < size; y++) {
39             printf("%d ", (chessBoard + x * size + y));
40         }
41         printf("\n");
42     }
43
44     // Membebaskan memori yang dialokasikan untuk papan catur
45     free(chessBoard);
46
47     return 0;
48 }

```

Terminal Output:

```

PS D:\alpro> cd "d:\alpro"; if ($?) { gcc dod.c -o dod }; if ($?) { .\dod }
Masukkan nilai i dan j (0 <= i, j < 8): 3 7
Papan Catur dengan Posisi yang Dapat Dicapai oleh Bidak Kuda:
0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 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

```

KOMPONEN PENILAIAN	YA	TIDAK
Soal 1 sesuai dengan output yang diinginkan	YA	
Soal 2 sesuai dengan output yang diinginkan	YA	
Bonus soal 1 dikerjakan	YA	

