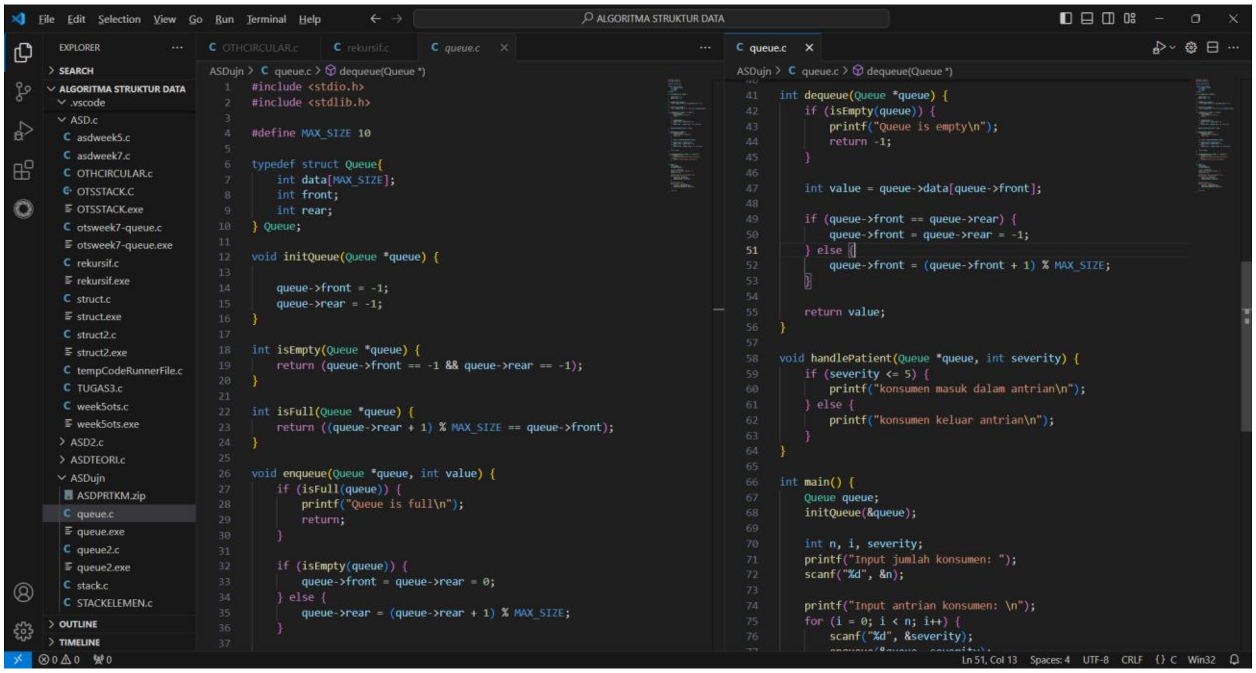
KODE UTP PRAKTIKUM ASD

NAMA:KEVIN BETESDA KORNELIUS BANGUN

NIM:1203230019

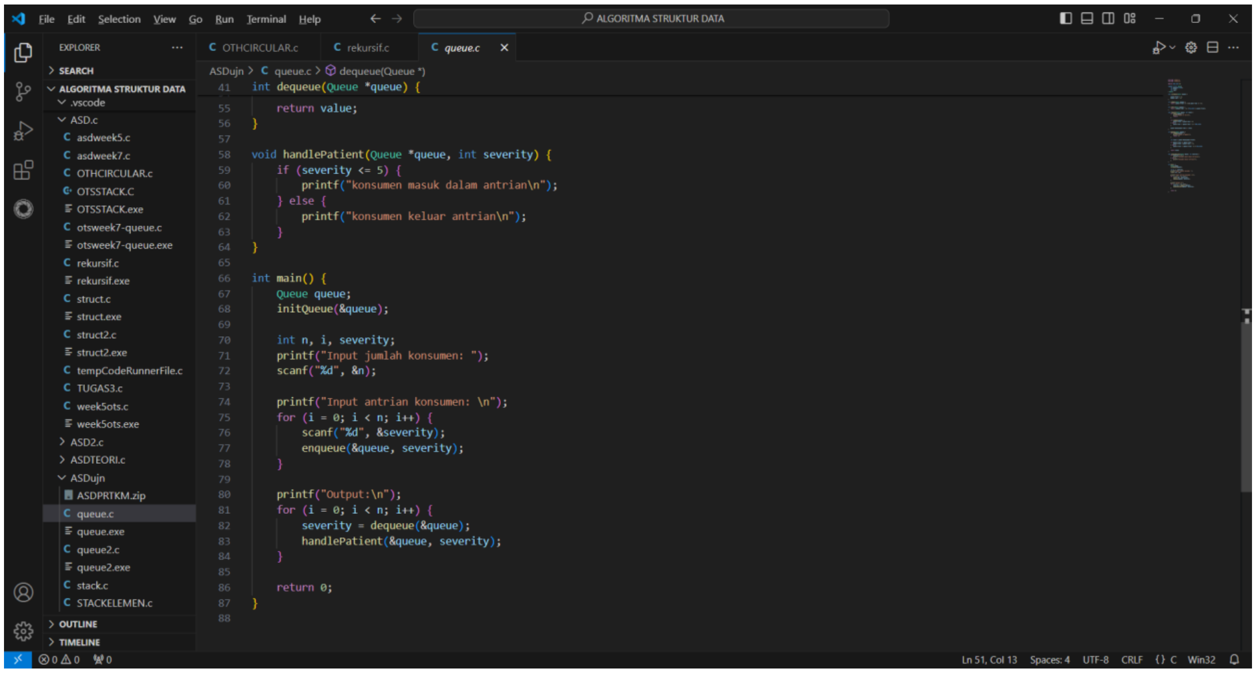
KELAS:IF 03-02

1.QUEUE



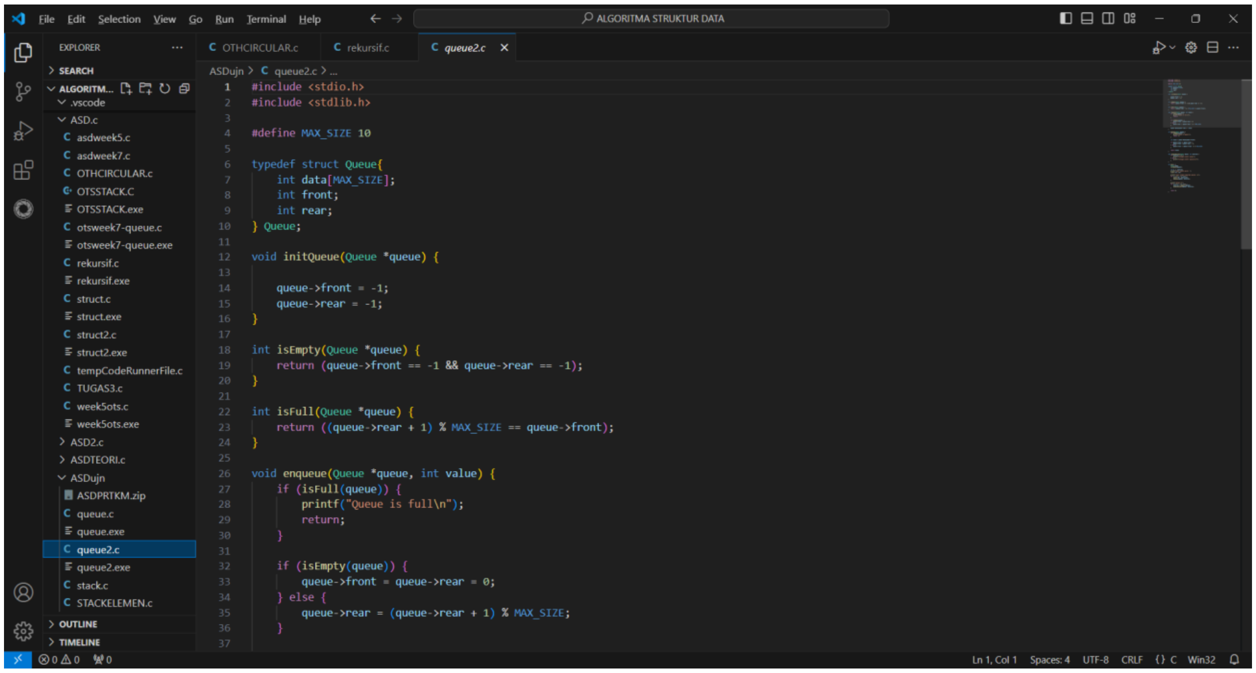
```c
#include <stdio.h>
#include <stdlib.h>

#define MAX_SIZE 10

typedef struct Queue{
    int data[MAX_SIZE];
    int front;
    int rear;
} Queue;

void initQueue(Queue *queue) {

    queue->front = -1;
    queue->rear = -1;
}

int isEmpty(Queue *queue) {
    return (queue->front == -1 && queue->rear == -1);
}

int isFull(Queue *queue) {
    return ((queue->rear + 1) % MAX_SIZE == queue->front);
}

void enqueue(Queue *queue, int value) {
    if (isFull(queue)) {
        printf("Queue is full\n");
        return;
    }

    if (isEmpty(queue)) {
        queue->front = queue->rear = 0;
    } else {
        queue->rear = (queue->rear + 1) % MAX_SIZE;
    }

```

```c
int dequeue(Queue *queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty\n");
        return -1;
    }

    int value = queue->data[queue->front];

    if (queue->front == queue->rear) {
        queue->front = queue->rear = -1;
    } else {
        queue->front = (queue->front + 1) % MAX_SIZE;
    }

    return value;
}

void handlePatient(Queue *queue, int severity) {
    if (severity <= 5) {
        printf("konsumen masuk dalam antrian\n");
    } else {
        printf("konsumen keluar antrian\n");
    }
}

int main() {
    Queue queue;
    initQueue(&queue);

    int n, i, severity;
    printf("Input jumlah konsumen: ");
    scanf("%d", &n);

    printf("Input antrian konsumen: \n");
    for (i = 0; i < n; i++) {
        scanf("%d", &severity);
```

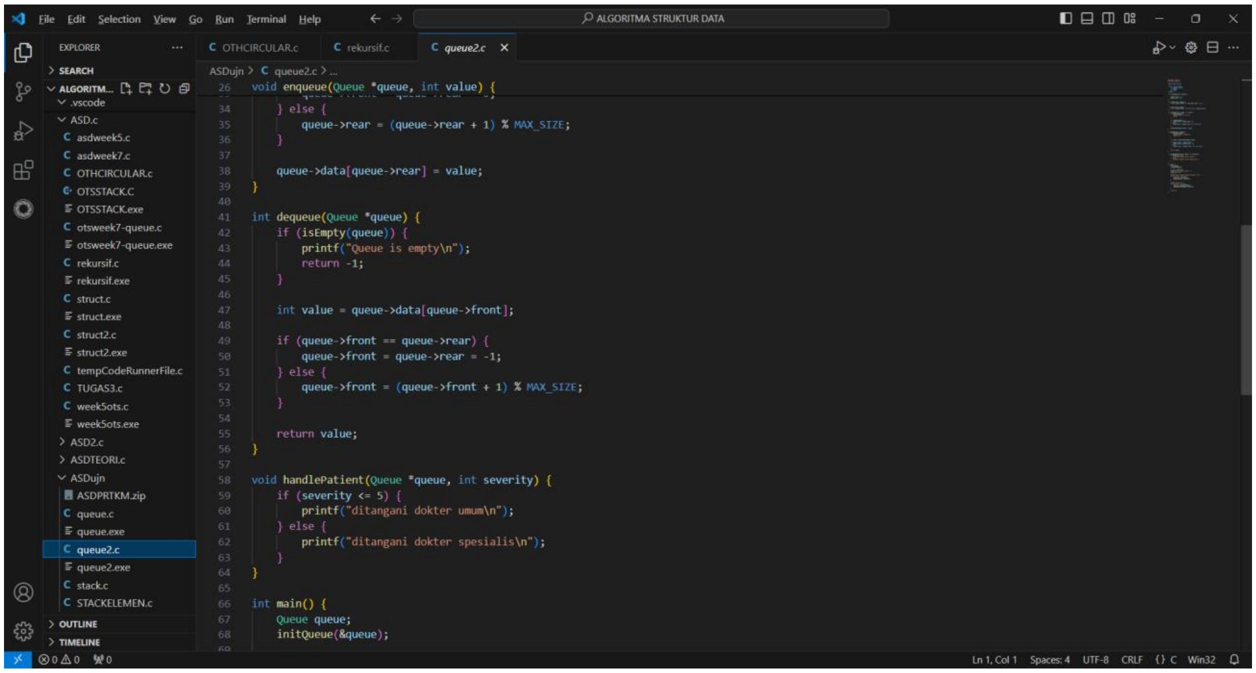


```c
int dequeue(Queue *queue) {
    return value;
}

void handlePatient(Queue *queue, int severity) {
    if (severity <= 5) {
        printf("konsumen masuk dalam antrian\n");
    } else {
        printf("konsumen keluar antrian\n");
    }
}

int main() {
    Queue queue;
    initQueue(&queue);

    int n, i, severity;
    printf("Input jumlah konsumen: ");
    scanf("%d", &n);

    printf("Input antrian konsumen: \n");
    for (i = 0; i < n; i++) {
        scanf("%d", &severity);
        enqueue(&queue, severity);
    }

    printf("Output:\n");
    for (i = 0; i < n; i++) {
        severity = dequeue(&queue);
        handlePatient(&queue, severity);
    }

    return 0;
}

```

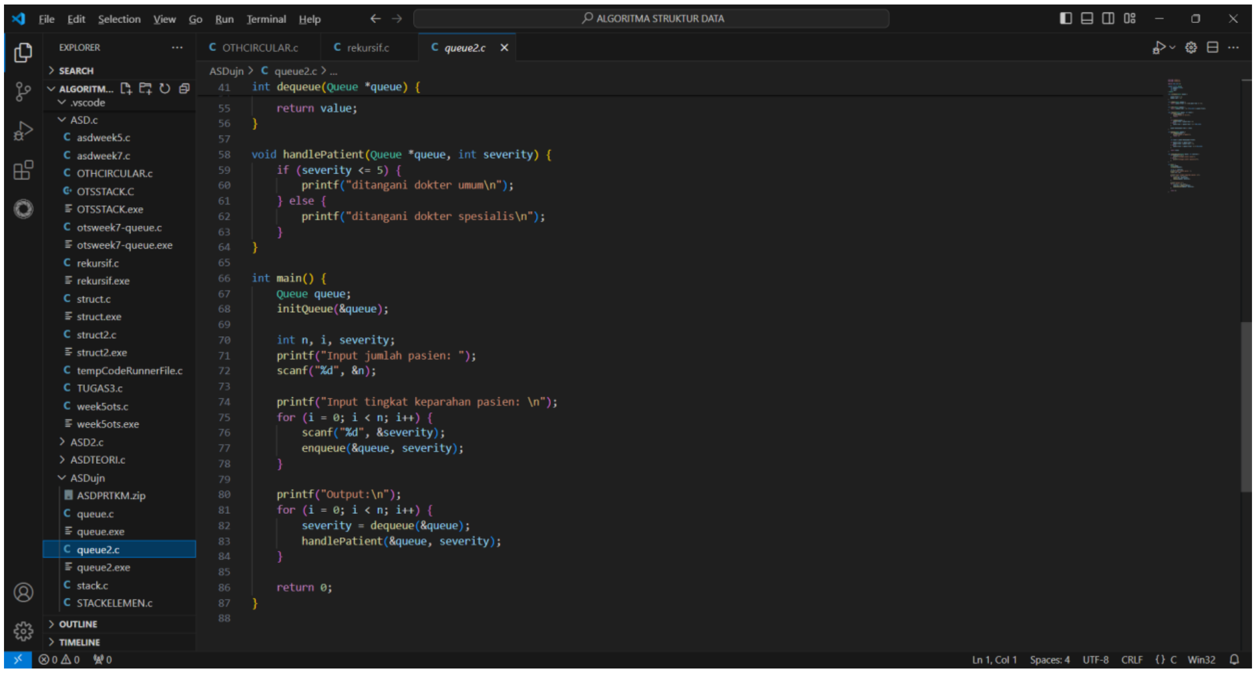
2.QUEUE



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

#define MAX_SIZE 10

typedef struct Queue{
    int data[MAX_SIZE];
    int front;
    int rear;
} Queue;

void initQueue(Queue *queue) {

    queue->front = -1;
    queue->rear = -1;
}

int isEmpty(Queue *queue) {
    return (queue->front == -1 && queue->rear == -1);
}

int isFull(Queue *queue) {
    return ((queue->rear + 1) % MAX_SIZE == queue->front);
}

void enqueue(Queue *queue, int value) {
    if (isFull(queue)) {
        printf("Queue is full\n");
        return;
    }

    if (isEmpty(queue)) {
        queue->front = queue->rear = 0;
    } else {
        queue->rear = (queue->rear + 1) % MAX_SIZE;
    }

```

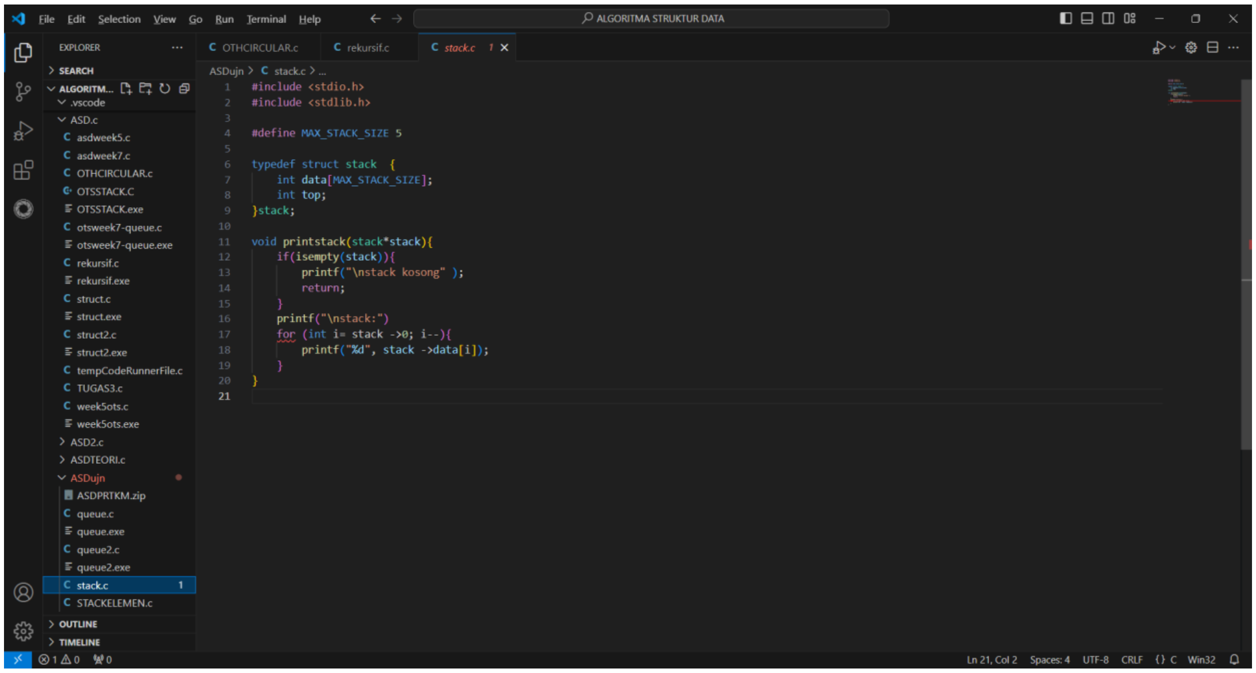


```c
void enqueue(Queue *queue, int value) {
    } else {
        queue->rear = (queue->rear + 1) % MAX_SIZE;
    }

    queue->data[queue->rear] = value;
}

int dequeue(Queue *queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty\n");
        return -1;
    }

    int value = queue->data[queue->front];

    if (queue->front == queue->rear) {
        queue->front = queue->rear = -1;
    } else {
        queue->front = (queue->front + 1) % MAX_SIZE;
    }

    return value;
}

void handlePatient(Queue *queue, int severity) {
    if (severity <= 5) {
        printf("ditangani dokter umum\n");
    } else {
        printf("ditangani dokter spesialis\n");
    }
}

int main() {
    Queue queue;
    initQueue(&queue);
```



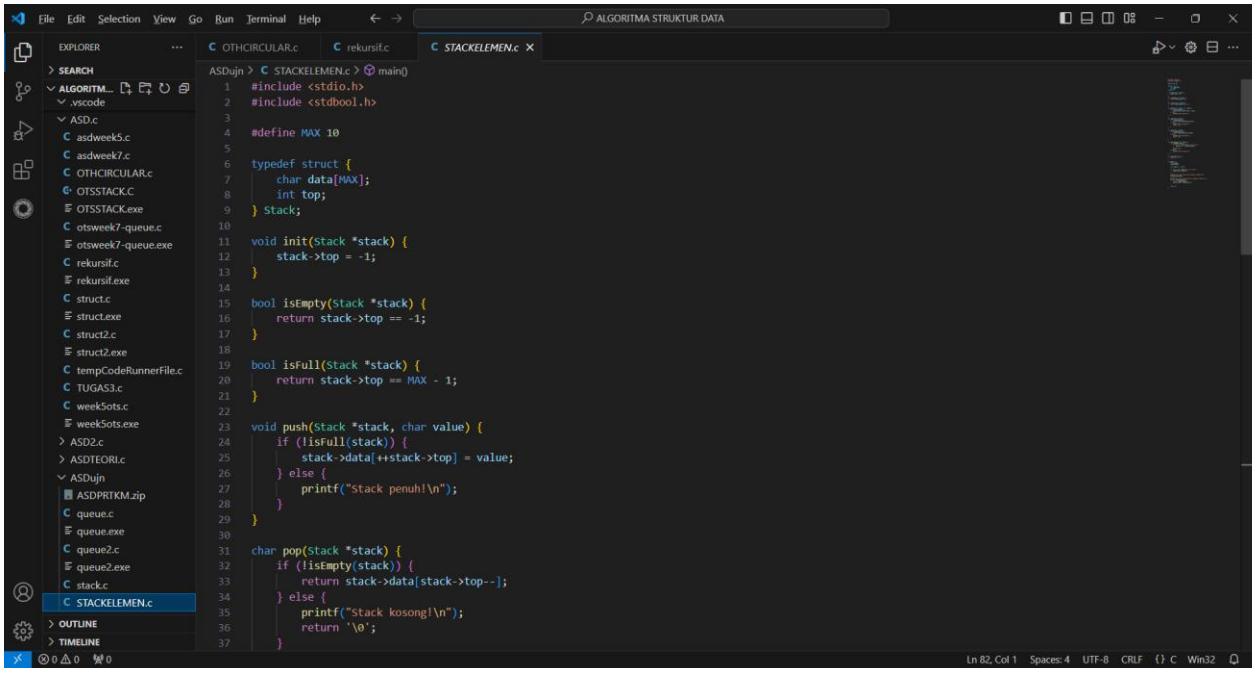
```c
int dequeue(Queue *queue) {
    return value;
}

void handlePatient(Queue *queue, int severity) {
    if (severity <= 5) {
        printf("ditangani dokter umum\n");
    } else {
        printf("ditangani dokter spesialis\n");
    }
}

int main() {
    Queue queue;
    initQueue(&queue);

    int n, i, severity;
    printf("Input jumlah pasien: ");
    scanf("%d", &n);

    printf("Input tingkat keparahan pasien: \n");
    for (i = 0; i < n; i++) {
        scanf("%d", &severity);
        enqueue(&queue, severity);
    }

    printf("Output:\n");
    for (i = 0; i < n; i++) {
        severity = dequeue(&queue);
        handlePatient(&queue, severity);
    }

    return 0;
}
```

3.STACK

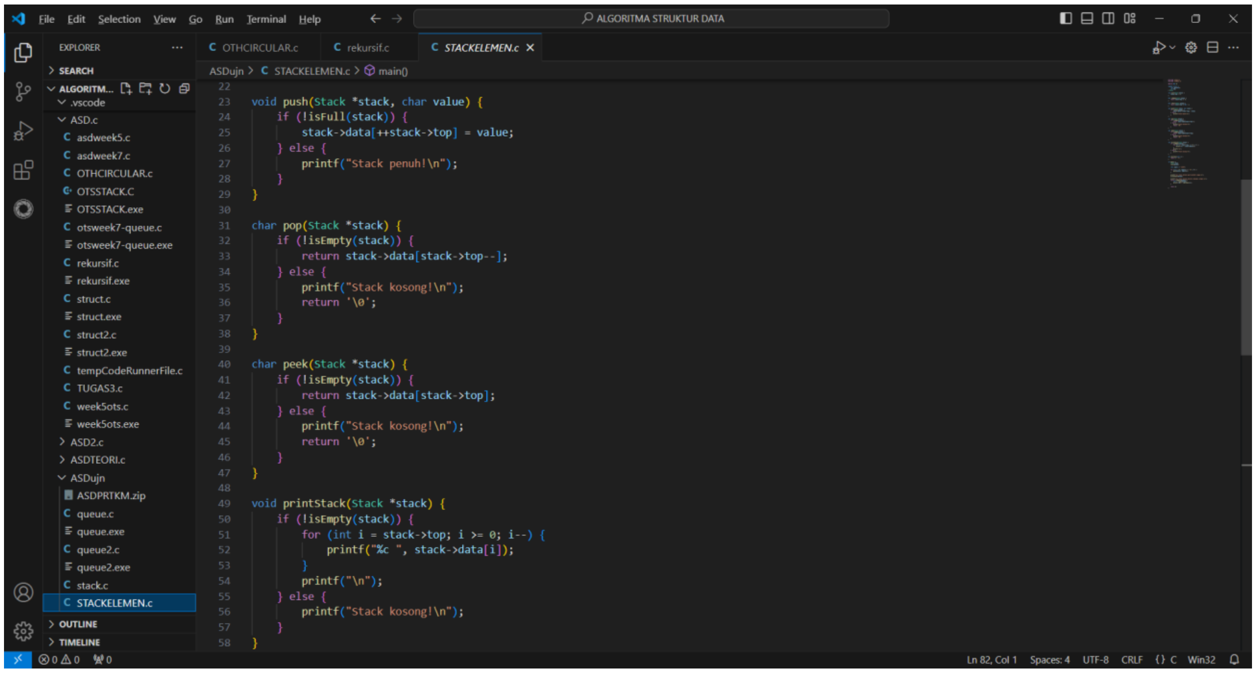


```c
#include <stdio.h>
#include <stdlib.h>

#define MAX_STACK_SIZE 5

typedef struct stack  {
    int data[MAX_STACK_SIZE];
    int top;
}stack;

void printstack(stack*stack){
    if(isempty(stack)){
        printf("\nstack kosong" );
        return;
    }
    printf("\nstack:")
    for (int i= stack ->0; i--){
        printf("%d", stack ->data[i]);
    }
}
```

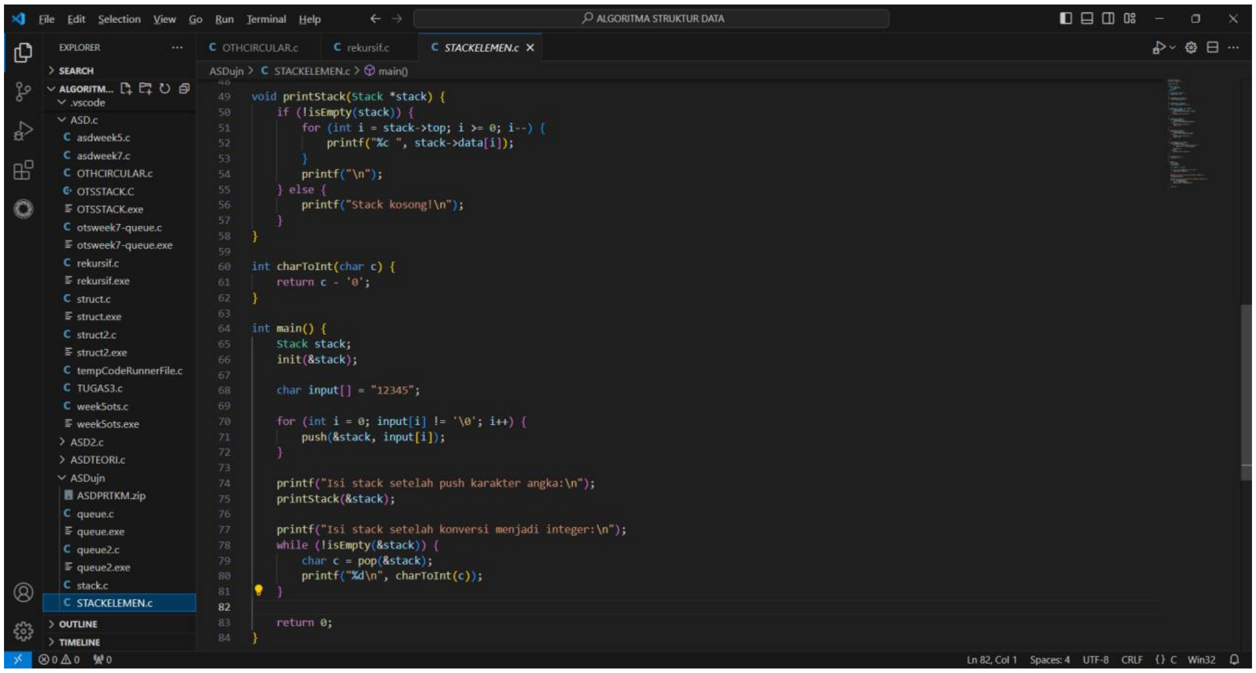
4.STACK ELEMEN



```c
#include <stdio.h>
#include <stdbool.h>

#define MAX 10

typedef struct {
    char data[MAX];
    int top;
} Stack;

void init(Stack *stack) {
    stack->top = -1;
}

bool isEmpty(Stack *stack) {
    return stack->top == -1;
}

bool isFull(Stack *stack) {
    return stack->top == MAX - 1;
}

void push(Stack *stack, char value) {
    if (!isFull(stack)) {
        stack->data[++stack->top] = value;
    } else {
        printf("Stack penuh!\n");
    }
}

char pop(Stack *stack) {
    if (!isEmpty(stack)) {
        return stack->data[stack->top--];
    } else {
        printf("Stack kosong!\n");
        return '\0';
    }
```


File Edit Selection View Go Run Terminal Help
ALGORITMA STRUKTUR DATA
EXPLORER
OTHCIRCULAR.c
rekursif.c
STACKELEMEN.c
ASDujn > STACKELEMEN.c > main()
Ln 82, Col 1 Spaces: 4 UTF-8 CRLF {} C Win32

```c

void push(Stack *stack, char value) {
    if (!isFull(stack)) {
        stack->data[++stack->top] = value;
    } else {
        printf("Stack penuh!\n");
    }
}

char pop(Stack *stack) {
    if (!isEmpty(stack)) {
        return stack->data[stack->top--];
    } else {
        printf("Stack kosong!\n");
        return '\0';
    }
}

char peek(Stack *stack) {
    if (!isEmpty(stack)) {
        return stack->data[stack->top];
    } else {
        printf("Stack kosong!\n");
        return '\0';
    }
}

void printStack(Stack *stack) {
    if (!isEmpty(stack)) {
        for (int i = stack->top; i >= 0; i--) {
            printf("%c ", stack->data[i]);
        }
        printf("\n");
    } else {
        printf("Stack kosong!\n");
    }
}
```



```c
void printStack(Stack *stack) {
    if (!isEmpty(stack)) {
        for (int i = stack->top; i >= 0; i--) {
            printf("%c ", stack->data[i]);
        }
        printf("\n");
    } else {
        printf("Stack kosong!\n");
    }
}

int charToInt(char c) {
    return c - '0';
}

int main() {
    Stack stack;
    init(&stack);

    char input[] = "12345";

    for (int i = 0; input[i] != '\0'; i++) {
        push(&stack, input[i]);
    }

    printf("Isi stack setelah push karakter angka:\n");
    printStack(&stack);

    printf("Isi stack setelah konversi menjadi integer:\n");
    while (!isEmpty(&stack)) {
        char c = pop(&stack);
        printf("%d\n", charToInt(c));
    }

    return 0;
}
```