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

#define N 10

typedef struct {
    int id;
    char descricao[100];
} Processo;

typedef struct {
    int ini;
    int fim;
    int cont;
    Processo vet[N];
} Fila;

Fila* cria_fila();
void incluir_processo(Fila* f, Processo p);
void retirar_processo(Fila* f);
void imprimir_fila(Fila* f);
int incr(int i);
void menu();

int main() {
    Fila* fila_de_processos = cria_fila();
    Processo p;
    int opcao;
    int proximo_id = 1;

    do {
        menu();
        scanf("%d", &opcao);

        switch (opcao) {
            case 1:
                if (fila_de_processos->cont >= N) {
                    printf("\nFila de processos cheia!\n\n");
                    break;
                }
                p.id = proximo_id++;
                printf("Digite a descricao do processo: ");
                while (getchar() != '\n');
                fgets(p.descricao, 100, stdin);
                p.descricao[strcspn(p.descricao, "\n")] = 0;

                incluir_processo(fila_de_processos, p);
                printf("Processo #%d (%s) incluido na fila.\n\n", p.id, p.descricao);
```

```
                break;
            case 2:
                retirar_processo(fila_de_processos);
                break;
            case 3:
                imprimir_fila(fila_de_processos);
                break;
            case 0:
                printf("Encerrando o sistema...\n");
                break;
            default:
                printf("Opcao invalida!\n\n");
        }
    } while (opcao != 0);

    free(fila_de_processos);
    return 0;
}

void menu() {
    printf("--- Sistema de Gerenciamento de Processos (Vetor) ---\n");
    printf("1. Incluir novo processo na fila\n");
    printf("2. Retirar processo (com maior tempo de espera)\n");
    printf("3. Imprimir fila de processos\n");
    printf("0. Sair\n");
    printf("Escolha uma opcao: ");
}

Fila* cria_fila() {
    Fila* f = (Fila*)malloc(sizeof(Fila));
    f->ini = 0;
    f->fim = 0;
    f->cont = 0;
    return f;
}

int incr(int i) {
    return (i + 1) % N;
}

void incluir_processo(Fila* f, Processo p) {
    if (f->cont < N) {
        f->vet[f->fim] = p;
        f->fim = incr(f->fim);
        f->cont++;
    }
}

void retirar_processo(Fila* f) {
```

```
    if (f->cont == 0) {
        printf("\nFila de processos esta vazia!\n\n");
        return;
    }

    Processo p_retirado = f->vet[f->ini];
    f->ini = incr(f->ini);
    f->cont--;

    printf("\nProcesso #%d (%s) foi retirado para execucao.\n\n", p_retirado.id,
p_retirado.descricao);
}

void imprimir_fila(Fila* f) {
    printf("\n--- FILA DE PROCESSOS ATUAL ---\n");
    if (f->cont == 0) {
        printf("Nenhum processo na fila.\n\n");
        return;
    }

    int i, pos = f->ini;
    for (i = 0; i < f->cont; i++) {
        printf("[ID: %d, Desc: %s] -> ", f->vet[pos].id, f->vet[pos].descricao);
        pos = incr(pos);
    }
    printf("FIM\n\n");
}
```