Produtores e Consumidores:

- O código deve incluir uma lógica onde uma ou mais threads produzem requisições (produtores) e outras processam essas requisições (consumidores).
- **Sincronização**:
    - Quando uma requisição é adicionada à fila, o **`pthread_cond_signal`** ou **`pthread_cond_broadcast`** notifica as threads consumidoras de que uma nova requisição está disponível.
    - As threads consumidoras esperam pela condição (**`pthread_cond_wait`**) até que uma requisição esteja disponível, assegurando que não fiquem ativas sem necessidade (busy-waiting).

Função da Thread Trabalhadora

1. A thread entra em um loop contínuo e tenta adquirir o **`mutex`** para acessar a fila.
2. Verifica se a condição de encerramento foi definida e se a fila está vazia; caso verdadeiro, libera o mutex e encerra.
3. Se a fila está vazia, entra em espera usando a variável de condição **`cond_requisicao`**, liberando temporariamente o **`mutex`** e aguardando novas requisições.
4. Quando uma requisição é encontrada, a thread a retira da fila, processa a operação correspondente e então continua o loop, pronta para processar a próxima requisição.
5. O loop se repete até que a condição de encerramento seja atendida.

Função adicionar_requisicao()

1. Bloqueia o acesso exclusivo à fila de requisições usando `mutex_fila`.
2. Verifica se o programa está encerrando (`encerrar`), em cujo caso libera o mutex e sai sem adicionar a requisição.
3. Gera um ID único para a nova requisição e define os parâmetros da operação (tipo de operação, conta de origem, conta de destino e valor).
4. Atualiza o índice `fim_fila` para apontar para o próximo espaço disponível na fila.
5. Sinaliza uma thread trabalhadora, indicando que uma nova requisição está disponível.
6. Libera o `mutex_fila`, permitindo que outras threads possam acessar a fila.

   A função **`void adicionar_requisicao(int operacao, int id_origem, int id_destino, float valor)`** é responsável por adicionar novas requisições à fila de operações que as threads trabalhadoras processarão. Essa função é chamada pelo servidor ou pelas threads clientes para inserir uma operação, como um depósito, transferência ou balanço, na fila.

- Atualização do Índice **`fim_fila`**:
  fim_fila = (fim_fila + 1) % MAX_REQUISICOES;
  A linha **`fim_fila = (fim_fila + 1) % MAX_REQUISICOES;`** incrementa **`fim_fila`** para apontar para o próximo espaço disponível na fila. O operador `%`

`MAX_REQUISICOES` assegura que `fim_fila` retorna ao início da fila (implementando uma fila circular) quando atinge o final da capacidade.

Função Thread cliente

1. **Inicia um Loop Contínuo** para gerar requisições enquanto `encerrar` for `0`.
2. **Gera uma Operação Aleatória** (depósito ou transferência) e seleciona contas de origem e destino, bem como o valor.
3. **Adiciona a Requisição à Fila** usando `adicionar_requisicao`, passando os parâmetros definidos para cada tipo de operação.
4. **Espera 1 segundo** antes de repetir o processo, controlando a taxa de novas requisições.
5. **Encerra a Função** quando `encerrar` for definido como `1`, indicando o fim do programa.

A função `void *cliente(void *arg)` é responsável por simular o comportamento de um cliente no sistema de servidor multithreaded. Cada cliente gera requisições aleatórias (como depósito ou transferência entre contas) e as adiciona à fila de requisições, para que possam ser processadas pelas threads trabalhadoras. Essa função é executada por threads clientes que representam vários usuários, simulando operações bancárias de forma concorrente.