ARGUMENTAÇÃO DE SEQUÊNCIA

```
AE: Baralho foi recebido
BAR_tpCondRet BAR_imprimeBaralho(BAR_tpBaralho baralho){

	PILHA_tpPilha auxiliar;
	CAR_tpCarta carta;

	if (baralho == NULL){
		return BAR_CondRetBaralhoNaoExiste;
	}
	AI1: Retorna BAR_CondRetBaralhoNaoExiste ou baralho recebido existe e não
é vazio, não fazendo nada.
	PILHA_criarPilha(&auxiliar);
	CAR_criarCarta(&carta);
	AI2: Baralho recebido existe e não é vazio.
	     Pilha e carta auxiliares vazias estão criadas.
	while (PILHA_verificaPilhaVazia(baralho->pBCartas) == PILHA_CondRetOK){
		PILHA_popPilha(baralho->pBCartas, &carta);
		CAR_imprimeCarta(carta);
		PILHA_pushPilha(auxiliar, carta);
	}
	return BAR_CondRetOK;
} /* Fim função: BAR Imprime Baralho */
AS: Todas as cartas do baralho foram impressas e excluídas do baralho.
    Baralho está vazio.
```

ARGUMENTAÇÃO DE REPETIÇÃO

```
AE: Baralho recebido existe e não é vazio.
    Pilha e carta auxiliares vazias estão criadas.
while (PILHA_verificaPilhaVazia(baralho->pBCartas) == PILHA_CondRetOK){
       PILHA_popPilha(baralho->pBCartas, &carta);
       CAR_imprimeCarta(carta);
       PILHA_pushPilha(auxiliar, carta);
}
AS: Todas as cartas do baralho foram impressas e excluídas do baralho.
    Baralho está vazio.
Ainv: Existem dois conjuntos: cartas a imprimir e cartas impressas.
       baralho->pBCartas aponta para a primeira carta do baralho.
```

1 – AE=>Ainv
Pela AE, baralho->pBCartas aponta para a primeira carta do baralho. Neste caso, todas as cartas estão no grupo de cartas a imprimir e o conjunto de cartas já impressas encontra-se vazio. Vale Ainv porque existem dois conjuntos e baralho->pBCartas aponta para a primeira carta do baralho.

2 – AE && (c == F) => AS
Pela AE o baralho recebido existe e não é vazio. Portanto como (c == F) todas as cartas do baralho foram impressas e excluídas do baralho. baralho->pBCartas aponta para NULL. Assim o baralho está vazio, valendo a AS.

3 – AE && (c == T) $\oplus$ B => Ainv
Pela AE, baralho->pBCartas aponta para a primeira carta do baralho. Como (c == T), essa primeira carta do baralho passa do conjunto cartas a imprimir para o conjunto cartas impressas, a mesma sai do baralho e baralho->pBCartas aponta para a próxima carta. Carta essa que agora é a primeira do baralho. Valendo assim Ainv.

4 - Ainv && (c == T) $\oplus$ B => Ainv
Para Ainv continuar válida, B precisa garantir que uma carta passe do conjunto cartas a imprimir para cartas impressas, essa mesma carta saia do baralho e baralho->pBCartas seja reposicionado.

5 – Ainv && (c == F) => AS
Com (c == F) todas as cartas já foram impressas e excluídas do baralho. Valendo assim AS.

6 – Término
O baralho contém um número finito de cartas a serem impressas. A cada ciclo uma carta é impressa e excluída e a próxima carta passa a ser a primeira do baralho. A repetição então termina após todas as cartas terem sido impressas, o baralho então está vazio, valendo AS.

ARGUMENTAÇÃO DE SEQUÊNCIA DO BLOCO DE REPETIÇÃO (WHILE)

```
AE: Baralho recebido existe e não é vazio.
    Pilha e carta auxiliares vazias estão criadas.
while (PILHA_verificaPilhaVazia(baralho->pBCartas) == PILHA_CondRetOK){
       PILHA_popPilha(baralho->pBCartas, &carta);
AI1: Carta no topo da pilha baralho foi obtida e retirada do baralho.
       CAR_imprimeCarta(carta);
AI2: Carta corrente foi impressa.
       PILHA_pushPilha(auxiliar, carta);
AI3: Carta corrente direcionada a uma pilha auxiliar.
}
AS: Todas as cartas do baralho foram impressas e excluídas do baralho.
    Baralho está vazio.
```

ARGUMENTAÇÃO DE SELEÇÃO

AE: Baralho foi recebido

```
if (baralho == NULL){
        return BAR_CondRetBaralhoNaoExiste;
}
```

AS: Retorna BAR_CondRetBaralhoNaoExiste ou baralho recebido existe e não é vazio, não fazendo nada.

1 – AE && (c == T) $\oplus$ B => AS

Pela AE, baralho foi recebido. Como (c==T) o baralho está vazio. Neste caso B retorna BAR_CondRetBaralhoNaoExiste e sai da função.

2 – AE && (c == F) => AS

Pela AE, baralho foi recebido. Como (c==F) o baralho existe e não é vazio. Valendo assim a AS.