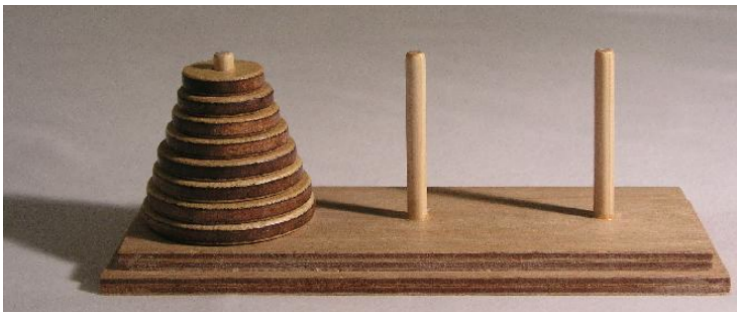
TORRE DE HANÓI

Neste trabalho prático, seu objetivo é construir uma versão do jogo Torre de Hanói implementando um tipo abstrato de dados TorreDeHanoi. Nele será usado um vetor de pinos de tamanho ***n,*** alocado dinamicamente, e ***m*** discos. Os valores de ***m*** e ***n*** serão definidos pelo usuário no início do programa.



1 - Entradas:

Entradas iniciais:

1. Um inteiro **n** (3 <= n <= 5) representando a quantidade de pinos.
2. Um inteiro **m** (3 <= m <= 5) representando a quantidade de discos.

Entradas sucessivas:

1. Um inteiro p1 (1 <= p1 <= n) representando o índice do pino no qual se deseja remover um disco.
2. Um inteiro p2 (1 <= p2 <= n) representando o índice do pino no qual deseja-se inserir o disco removido do pino anterior.

As entradas se encerram quando o jogador move todos os discos para um único pino diferente do pino 1.

2 - Saída:

1. Caso a quantidade de pinos inserida pelo usuário esteja fora do intervalo especificado, seu programa deve imprimir a mensagem de erro “Entrada invalida” e solicitar uma nova entrada (Figura 1).
2. Caso a quantidade de discos inserida pelo usuário esteja fora do intervalo especificado, seu programa deve imprimir a mensagem de erro “Entrada invalida” e solicitar uma nova entrada (Figura 1).
3. Antes de cada jogada, o programa deve imprimir os pinos com os discos (Figura 2).
4. Cada disco de tamanho ***k*** é representado por ***k*** underlines (_) na esquerda e na direita de uma barra vertical (|). Os (_) representam os discos e as (|) representam os pinos (Figura 2).

5. Abaixo dos pinos deve ser impresso uma base com underlines e barras verticais conforme as imagens abaixo (Figura 2).
6. A largura de cada base é de **2m + 3** caracteres conforme as imagens abaixo. (Figura 2).
7. Abaixo das bases deve-se imprimir uma linha em branco. Após isso, deve-se imprimir em uma linha o índice de cada pino (Figura 2).
8. No início e no final da impressão deve haver uma linha em branco (Figura 2).
9. Após cada jogada, os pinos devem ser atualizados e impressos novamente (Figura 3).
10. Caso o jogador insira uma entrada fora do intervalo dos pinos, ou tente remover um disco de um pino vazio, ou tente colocar um disco maior por cima de um disco menor, o programa deverá imprimir a mensagem de erro “Movimento inválido” e solicitar uma nova entrada (Figura 4).
11. Quando o jogador conseguir colocar todos os discos em um único pino, diferente do pino 1, o programa deverá imprimir a mensagem “PARABENS VOCE CONSEGUIU” seguido da mensagem “TOTAL DE JOGADAS: x” onde ***x*** representa a quantidade de movimentações (Figura 5).

```
Insira a quantidade de pinos: [3..5] 1
Entrada invalida
Insira a quantidade de pinos: [3..5] 3
Insira a quantidade de discos: [3..5] 6
Entrada invalida
Insira a quantidade de discos: [3..5] 5
```

Figura 1.

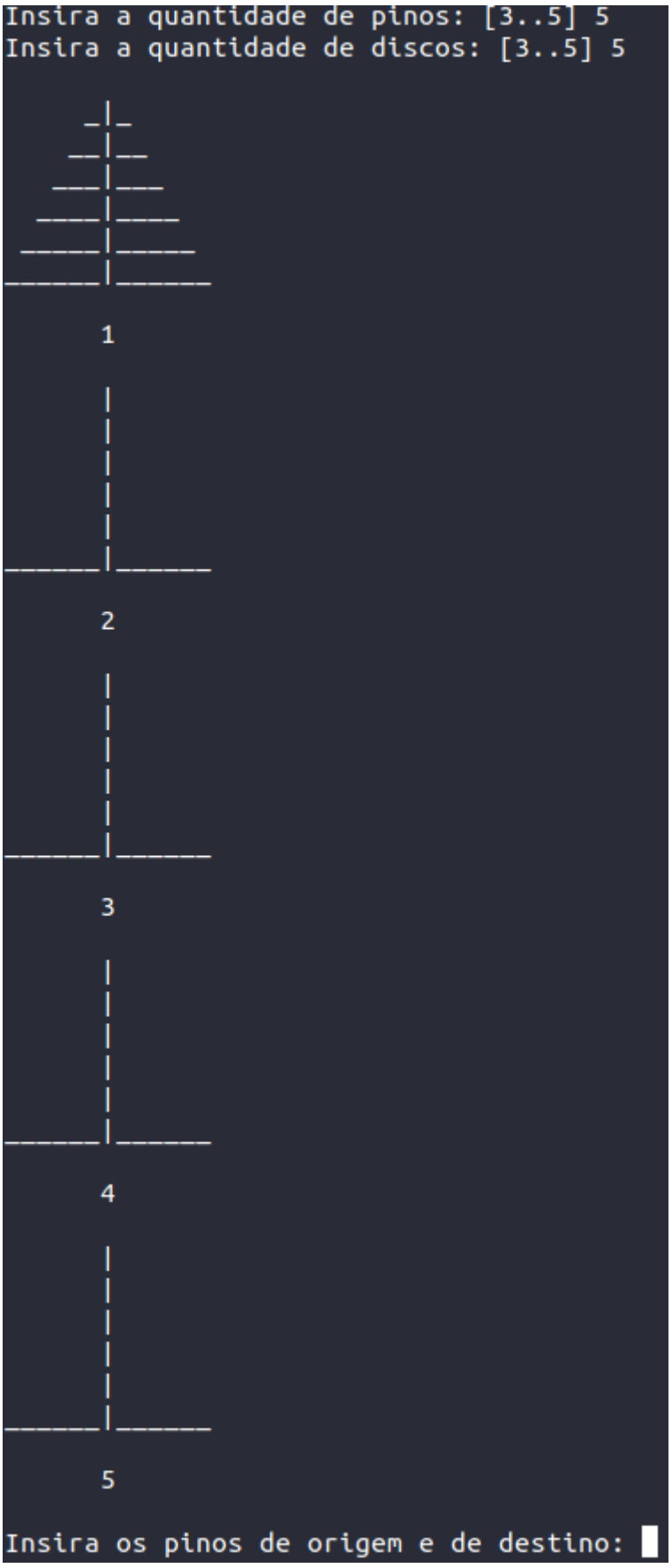

Insira a quantidade de pinos: [3..5] 5
Insira a quantidade de discos: [3..5] 5
1
2
3
4
5
Insira os pinos de origem e de destino:

Figura 2.

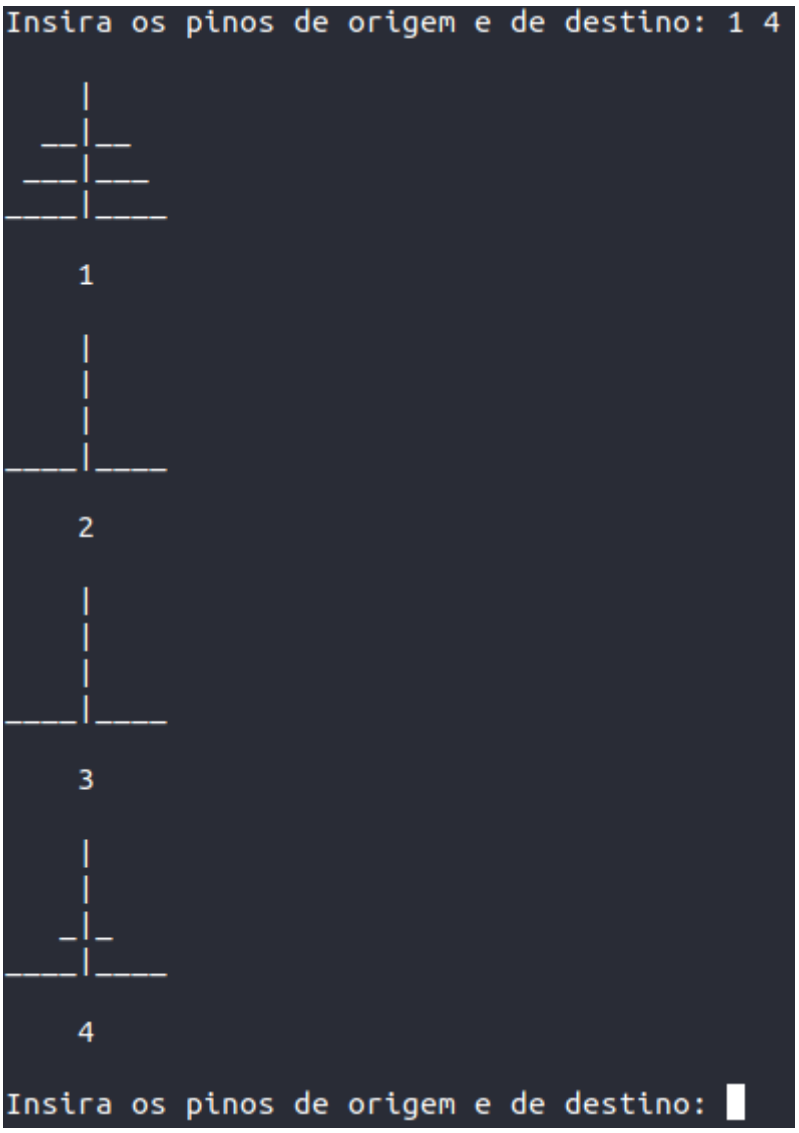

Insira os pinos de origem e de destino: 1 4
1
2
3
4
Insira os pinos de origem e de destino:

Figura 3.

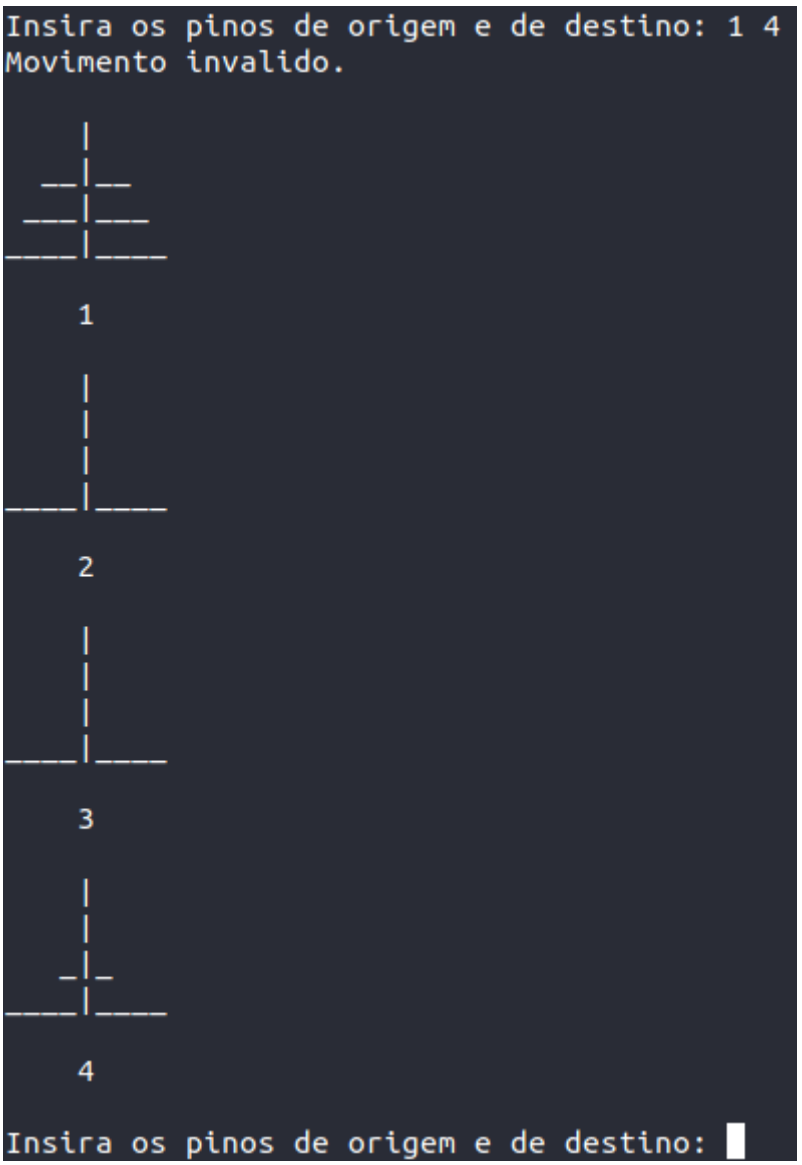

Insira os pinos de origem e de destino: 1 4
Movimento invalido.
1
2
3
4
Insira os pinos de origem e de destino:

Figura 4.

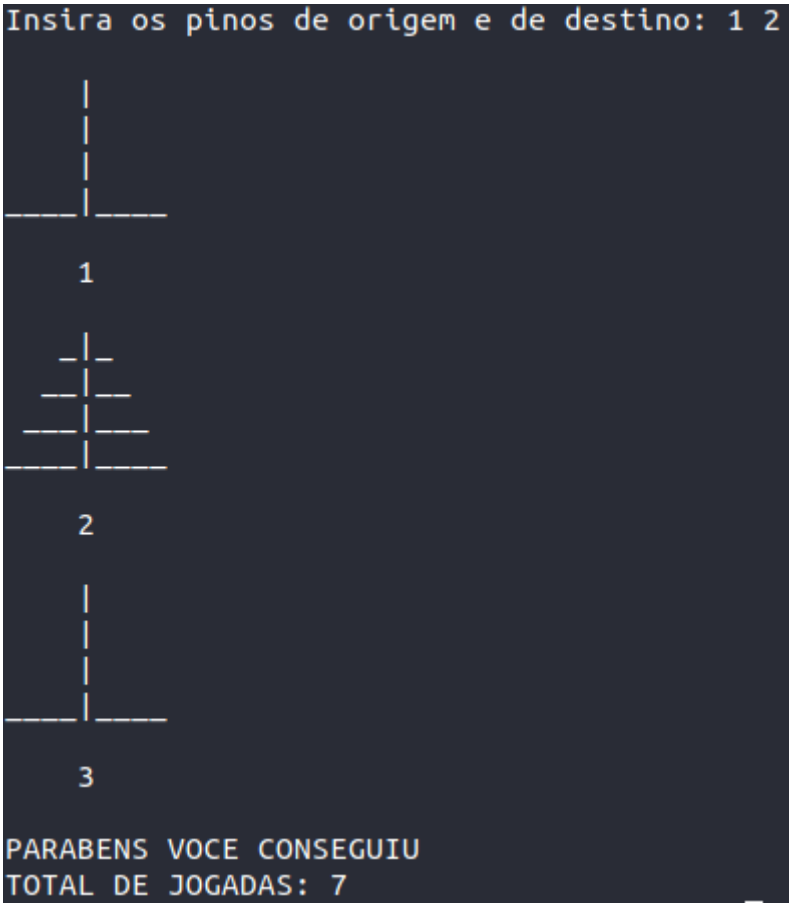


Figura 5.

3 - Observações:

1. O jogo deve começar com todos os discos ordenados no pino 1.
2. É permitido criar funções adicionais, mas a lógica de início do jogo deve ser resolvida na função main.
3. Todos os protótipos listados devem ser utilizados e **não podem ser modificados.**
4. Não se esqueça de **desalocar** todos os pinos e discos ao final do programa.
5. Todas as variáveis utilizadas **devem ter nomes significativos** e **todas as instruções devem ter comentários** explicando o funcionamento do código. Evite comentários supérfluos como: “Essa linha soma duas variáveis”.
6. Nesta atividade as seguintes structs **devem ser utilizadas**:

```
struct disco{
   Disco *next; //ponteiro para o proximo disco
   char tamDisco; //tamanho do disco
};
```

```
struct pino{
   Disco *topo; //ponteiro para o topo da pilha
   char numDiscos; //quantidade de discos
};
```

4 - Funções necessárias. Estas funções devem unicamente executar o que foi descrito.

1. criarPinos: Deverá receber como parâmetro um inteiro ***n*** representando a quantidade de pinos e retornar um vetor do tipo Pino alocado dinamicamente. Utilize o protótipo: ***Pino** criarPinos(int n);***
2. moverDisco: Deverá receber como parâmetro um vetor ponteiro para pinos, o índice do pino no qual deseja-se remover um disco (pinoOrigem) e o índice do pino no qual deseja-se colocar o disco removido (pinoDestino). Se não for possível fazer a movimentação a função deve retornar 0, caso contrário realiza os movimentos dos discos e retorna 1.
3. Na função moverDisco, a movimentação deve ser feita alterando os ponteiros no campo ***next***, não é permitido alocar ou desalocar novos discos. Utilize o protótipo: ***int moverDisco(Pino **pinos, int pinoOrigem, int pinoDestino);***
4. imprimir: Deverá receber como parâmetro um vetor de Pino (pinos), a quantidade de pinos (numPinos) e a quantidade de discos (numDiscos) e imprimir todos os pinos e discos conforme as figuras acima. Nesta função **é proibido utilizar vetores auxiliares**. Utilize o protótipo: ***void imprimir(Pino **pinos, int numPinos, int numDiscos);***
5. criarPino: Deverá retornar um pino vazio. Utilize o protótipo: **Pino* criarPino();**
6. criarDisco: Recebe como parâmetro o tamanho de um disco (tam), e retornar um disco de tamanho tam. Utilize o protótipo: ***Disco* criarDisco(int tam);***
7. pop: Recebe como parâmetro o endereço de um pino, desempilha o disco que está no topo e retorna seu endereço. Utilize o protótipo: ***Disco* pop(Pino *pino);***
8. push: Recebe como parâmetro o endereço de um pino e o endereço de um disco e empilha o disco no pino. Utilize o protótipo: ***void push(Pino *pino, Disco *disco);***
9. excluirPino: Recebe o endereço de um pino e desaloca o pino e todos os seus discos. Utilize o protótipo: ***void excluirPino(Pino *pino);***

5 - Protótipos TorreDeHanoi.h:

1. Pino* criarPino();
2. Disco* criarDisco(int tam);
3. Disco* pop(Pino *pino);
4. void push(Pino *pino, Disco *disco);
5. void excluirPino(Pino *pino);

6 - Especificações de envio:

Os três arquivos abaixo devem ser compactados e submetidos no formato zip.

1. TorreDeHanoi.h: deve conter **unicamente** todos os protótipos de funções especificados na seção 5 e structs especificadas na seção 3. Todos os protótipos devem ter comentários explicando os parâmetros que a função recebe e o que ela faz.
2. TorreDeHanoi.c: **deve conter unicamente** as implementações das funções do arquivo TorreDeHanoi.h .
3. main.c: deve conter a implementação do jogo e as demais funções.