

Trabalho 2: Paciência Spider

Nome: Daniel Siles – 1311291

Fernando Homem – 1211971

Mateus Castro – 1213068

Informações para o modelo:

Exigências do ENUNCIADO:

1) As cartas de um baralho começam a ser divididas em dez sequências em baralhadas com os valores e naipes visíveis de apenas uma única carta e mais cinco montes.

2) O trabalho deverá ter um módulo "pilha de cartas" a ser utilizada pelos os módulos baralho inicial, em cada uma das 10 sequências principais (mostram apenas a última carta) administrados por outro módulo, nos cinco montes (outro módulo) e nas sequências finais (outro módulo).

3) Deverá ter módulo carta que representa uma carta.

4) 4 níveis de dificuldade:

1 naipe (fácil),

2 naipes (médio),

3 naipes (difícil),

4 naipes (muito difícil).

5) O módulo "pilha de cartas" deverá utilizar a estrutura lista duplamente encadeada genérica com cabeça devidamente encapsulada em um módulo que será acoplado a outro.

6) Uma estrutura para cada jogo. Esta será uma lista de listas, cada nó desta lista de listas apontará para o cabeça de uma lista necessária, ou seja, cada nó apontará para lista da sequência¹, outro para a lista da sequência², etc..., outro apontará para lista de monte.

Carta:

- .Visível ou Escondida
- .Naipes
- .Numeração

Baralho:

- .52 cartas
- .4 Naipes
- .13 cartas por Naipes

Sequência de 5 cartas (Monte de 5):

- .Máximo de 5 cartas de mesmo ou diferentes naipes
- .Pode ser sequência vazia
- .Primeira carta é virada para cima e as outras para baixo
- .São 6 sequências de 5 cartas

Sequência de 6 cartas (Monte de 6):

- .Máximo de 6 cartas de mesmo ou diferentes naipes
- .Pode ser sequência vazia
- .Primeira carta é virada para cima e as outras para baixo
- .São 4 sequências de 6 cartas

Sequência de 10 cartas (Morto):

- .São 10 cartas de mesmo ou diferentes naipes
- .Não pode ser sequência vazia
- .Todas viradas para baixo
- .São 5 sequências de 10 cartas

Sequência de 13 cartas (Finalizada):

- .São 13 cartas de mesmo naipe
- .Não pode ser sequência vazia
- .É uma sequência ordenada (A,2,3,4,5,6,7,8,9,10,J,Q,K)

Baralho Inicial:

- .São 104 cartas de mesmo ou diferentes naipes
- .Não pode ser uma sequência vazia
- .Não precisa ser ordenada
- .Precisa ser embaralhada

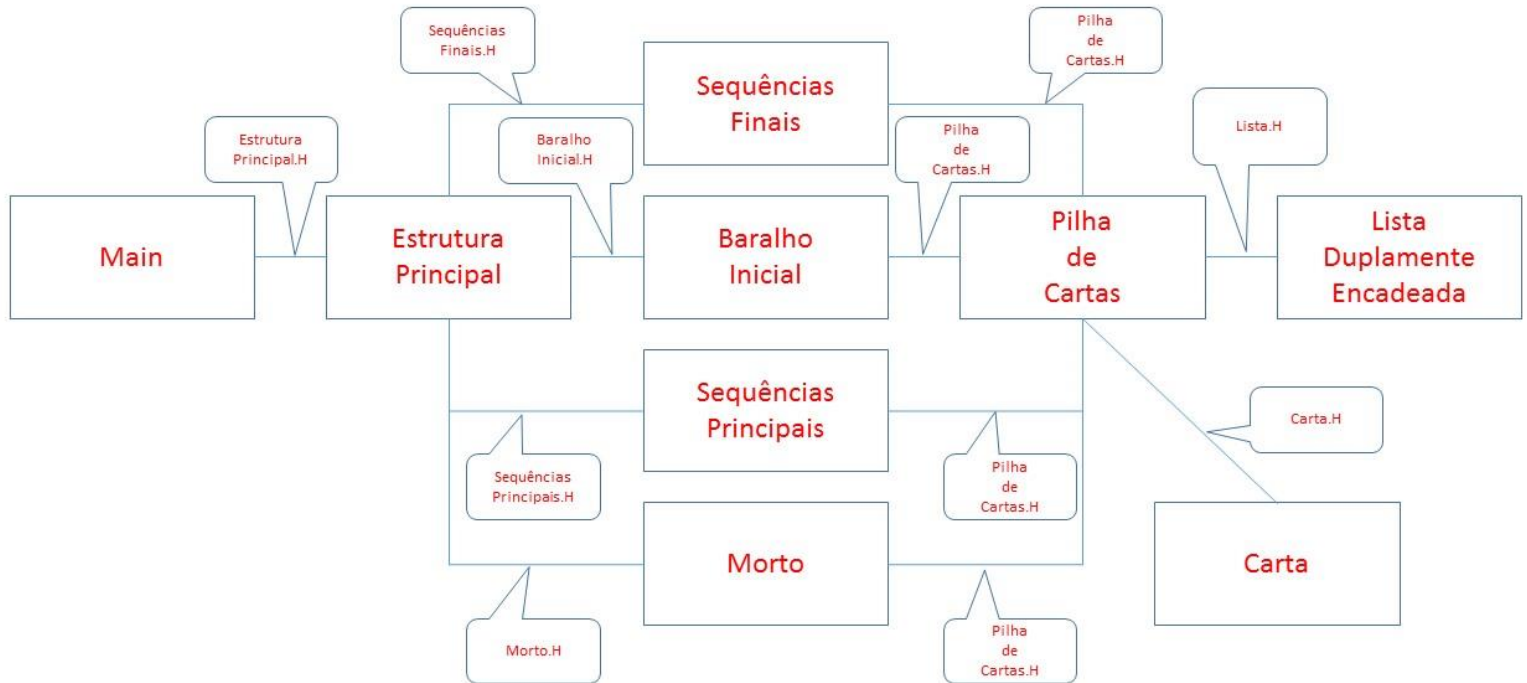
Estrutura Principal

- .Novo Jogo-> Começar um jogo novo.
- .Continuar/Carregar Jogo -> continuar um jogo já salvo.
- .Salvar-> Salvar jogo corrente.
- .Dificuldade->Selecionar o nível de dificuldade do Jogo.
(Fácil – 1 Naipes, Médio – 2 Naipes, Difícil – 3 Naipes, Muito Difícil – 4 Naipes)
- .Sair-> fechar a aplicação.

Modelo Estrutural:



Diagrama de Arquitetura



Módulos:

1) Carta.H/Carta.C

```
. CAR_tpCondRet CAR_criarCarta(CAR_tpCarta *pCarta);  
. CAR_tpCondRet CAR_destruirCarta(CAR_tpCarta pCarta);  
. CAR_tpCondRet CAR_editarCarta(CAR_tpCarta pCarta, char NovaFace, char  
NovoNaipes, char NovaPosicao);  
.CAR_tpCondRet CAR_retornaPosicao (CAR_tpCarta pCarta, char* pPosicao);  
. CAR_tpCondRet CAR_retornaNaipes (CAR_tpCarta pCarta, char* pNaipes);  
. CAR_tpCondRet CAR_retornaFace(CAR_tpCarta pCarta, char* pFace);  
. CAR_tpCondRet CAR_imprimeCarta(CAR_tpCarta pCarta);
```

2)LISTA.H/LISTA.c

```
. LIS_tpCondRet LIS_CriarLista(LIS_tppLista * pLista, void(*ExcluirValor) (void *  
pDado));  
. LIS_tpCondRet LIS_DestruirLista(LIS_tppLista pLista);  
. LIS_tpCondRet LIS_EsvaziarLista(LIS_tppLista pLista);  
. LIS_tpCondRet LIS_InserirElementoAntes(LIS_tppLista pLista, void * pValor);  
. LIS_tpCondRet LIS_InserirElementoApos(LIS_tppLista pLista, void * pValor);  
. LIS_tpCondRet LIS_ExcluirElemento(LIS_tppLista pLista);  
. LIS_tpCondRet LIS_ObterValor(LIS_tppLista pLista, void ** pValor);  
. LIS_tpCondRet LIS_IrInicioLista(LIS_tppLista pLista);  
. LIS_tpCondRet LIS_IrFinalLista(LIS_tppLista pLista);  
. LIS_tpCondRet LIS_AvancarElementoCorrente(LIS_tppLista pLista, int numElem);  
. LIS_tpCondRet LIS_ProcurarValor(LIS_tppLista pLista, void * pValor);  
. LIS_tpCondRet LIS_retornaNumElementos(LIS_tppLista pLista);
```

3)PilhadeCartas.H/PilhadeCartas.C

```
. PILHA_tpCondRet PILHA_criarPilha(PILHA_tpPilha *pPilha);  
. PILHA_tpCondRet PILHA_pushPilha(PILHA_tpPilha pPilha, CAR_tpCarta c);  
. PILHA_tpCondRet PILHA_popPilha(PILHA_tpPilha pPilha, CAR_tpCarta*pCarta);  
. PILHA_tpCondRet PILHA_liberaPilha(PILHA_tpPilha pPilha);  
. PILHA_tpCondRet PILHA_verificaPilhaVazia(PILHA_tpPilha pPilha);  
. PILHA_tpCondRet PILHA_imprimePilha(PILHA_tpPilha pPilha);
```

4)BaralhoInicial.H/BaralhoInicial.C

```
. BAR_tpCondRet BAR_criarBaralho(BAR_tpBaralho *pBaralho);  
. BAR_tpCondRet BAR_liberarBaralho(BAR_tpBaralho pBaralho);
```

```
. BAR_tpCondRet BAR_inicializarBaralho(BAR_tpBaralho pBaralho, int numNaipes);  
. BAR_tpCondRet BAR_embaralharBaralho(BAR_tpBaralho pBaralho);  
. BAR_tpCondRet BAR_imprimeBaralho(BAR_tpBaralho baralho);
```

5) Sequências Principais.H/

Sequências Principais.C

```
. SQP_tpCondRet SQP_criarSequencia(SQP_tpSQPrincipal *SQPrincipal);  
. SQP_tpCondRet SQP_liberaSequencia(SQP_tpSQPrincipal sqPrincipal);  
. SQP_tpCondRet SQP_removeDaSequencia(SQP_tpSQPrincipal sqTira, CAR_tpCarta  
cCarta, PILHA_tpPilha *pPilhaGuarda);  
. SQP_tpCondRet SQP_adicionaNaSequencia(PILHA_tpPilha pPilhaTira,  
SQP_tpSQPrincipal sqRecebe);  
. SQP_tpCondRet SQP_verificaSequenciaCompleta(SQP_tpSQPrincipal SQP);
```

6) Sequências Finais.H/

Sequências Finais.C

```
. SQF_tpCondRet SQF_criarSQFinal(SQF_tpSQFinal *SQFinal);  
. SQF_tpCondRet SQF_liberaSQFinal(SQF_tpSQFinal SQFinal);  
. SQF_tpCondRet SQF_inicializaSQFinal(SQF_tpSQFinal SQFinal, PILHA_tpPilha sqFinal);
```

7) Morto.H/Morto.C

```
. MOR_tpCondRet MOR_criarMorto(MOR_tpMorto *mMorto, PILHA_tpPilha pPilha);  
. MOR_tpCondRet MOR_liberaMorto(MOR_tpMorto mMorto);  
. MOR_tpCondRet MOR_popMorto(MOR_tpMorto mMorto, CAR_tpCarta *cartaPop);
```

8) Estrutura Principal.H/

Estrutura Principal.C

```
. IniciaNovoJogo()  
. CarregaJogoSalvo()  
. SalvaJogo()  
. EscolheDificuldade()  
. SaiDoJogo()  
. RealizaJogada()  
. DistribuiBaralho()  
. DistribuiMorto
```