Resumo Trabalho Final de Estrutura de Dados.

Alunos: Bruno Torres Marques e Jhonattan Nascimento Barbosa

Prof: Enyo

O código possui 5 arquivos sendo: dois arquivos .h com os protótipos de cada uma das funções e structs utilizadas, 2 arquivos .c onde são implementadas os arquivos implementados da árvore e das funções de jogo e por fim 1 arquivo main.c para rodar o programa como um todo.

No arquivo “arvore.c” são implementadas duas structs Arvore e No e as funções de inserção, criação, remoção e liberar da árvore binaria e as funções que fariam o balanceamento avl(funções de rotação, fator de balanceamento, calcular altura dos nós, além de inserção e remoção modificadas) mas devido a erros de compilação não conseguimos colocá-las para funcionamento, mesmo assim estamos enviando doa-as para análise. As funções da árvore binaria permanecem iguais as disponibilizadas pelo professor, foi modificado apenas as condições de remoção. Modificamos a função para que ela sempre remova o menor valor como necessário para o jogo.

No arquivo jogo.h além de termos os protótipos das funções e temos a declaração da struct jogador com seus campos nome, deck, rodadas e partidas. No arquivo “jogo.c” temos uma struct carta e mais 7 funções todas relacionadas ao funcionamento do jogo. A struct carta possui 13 contadores, onde cada um e responsável por uma carta do baralho e mais tarde serão usados para comparação ao inserir. Temos a função criar jogador() na qual e alocado espaço de memoria para uma struct jogador, inserido seu nome, e inicializando os contadores com zero, no campo deck da struct recebe-se a função preencher árvore passando o ponteiro struct carta como parâmetro. Logo após finalizado a struct jogador e totalmente retornada para main onde a real struct já recebe tudo pronto.

Na função criar car() é alocado espaço de memoria para um ponteiro struct do tipo Carta e setado a todos os seus campos o valor como 0. A função preencher arvore() gera os valores aleatórios e além de verificar se foram inseridas mais do que 4 cartas da mesma nos decks. A função jogar é a que realiza as remoções e compara quais foram os ganhadores da rodada pontuando o contador da maior carta retirada de seu respectivo baralho. A função ganhou_partida() verifica quem ganhou mais rodadas e pontua para o o campo de partidas do jogador que mais pontuou no contador de rodadas. Por fim temos mais duas funções vencedor jogo() no qual verifica qual foi o jogador que pontuou o maior número de partidas e printa o/os jogadores que ganharam o jogo, além da função liberar_jog() na qual libera o espaço alocado por jogador e o espaço alocado pela árvore.

Por fim temos a main no qual instanciamos o srand para garantir os valores aleatórios, os três ponteiros jogadores, a struct carta. Primeiramente lemos a quantidade de partidas que serão jogados pelos três jogadores, criamos cada um deles e preenchemos seus respectivos decks para cada interação de do while até as partidas serem acabadas e chegarmos ao resultado do ganhador do jogo.