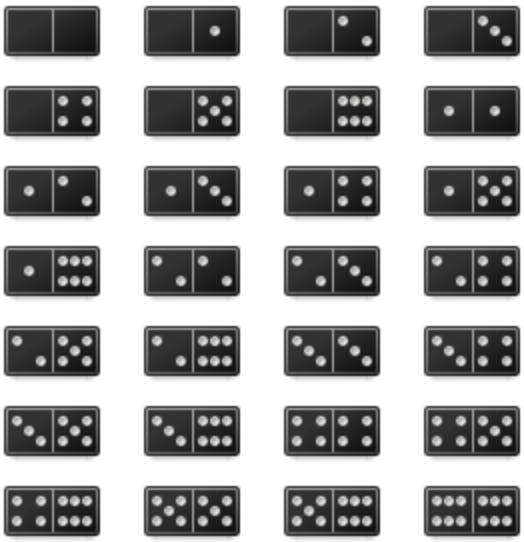
Universidade Regional Integrada do Alto Uruguai e das Missões
Departamento de Engenharias e Ciência da Computação
Curso de Ciência da Computação
Linguagem de Programação II

Documentação do Jogo de Dominó

Raffael Bottoli Schemmer
Amilcar Bittencourt
Henrique Maurer
Roni Bigolin

Descrição do Jogo

O jogo basicamente é formado por 28 pedras numeradas de zero a seis respectivamente. Abaixo as pedras do jogo de dominó :



Regras:

- Cada jogador recebe 7 peças
- O jogo deve possuir no minimo 2 e no máximo 4 jogadores
- Quem possuir a maior peca de lados iguais começa jogando. Caso nenhum dos jogadores possuir pedras com lados iguais, todos da mesa devem somar os valores dos lados de cada pedra e qual o jogador tiver o valor do somatório dos lados começa jogando esta pedra na mesa.
- A seqüência de jogadas acontece no sentido anti-horário do primeiro a jogar.
- Jogar uma pedra significa ler da mão procurando pedras com lados iguais aos do tabuleiro e jogar tais ao tabuleiro.
- Caso a mão não conter lados iguais ao tabuleiro o jogador deve se dirigir ao monte e buscar peca por peca verificando se tal contem um dos lados iguais aos do tabuleiro.
- Caso o monte seja lido até o fim e o jogador não encontre uma pedra valida, ele deve passar a vez para que os jogadores continuem jogando suas pedras.
- Ganha o jogo quem não tiver nenhuma pedra na mão
- Se acontecer a exceção de unanimidade dos jogadores não terem mais pedras para jogar na mesa e o monte estiver vazio, soma-se o numero de pedras da mão de cada jogador e quem conter o menor numero de pedras ganha o jogo.

Conhecimento dos jogadores virtuais :

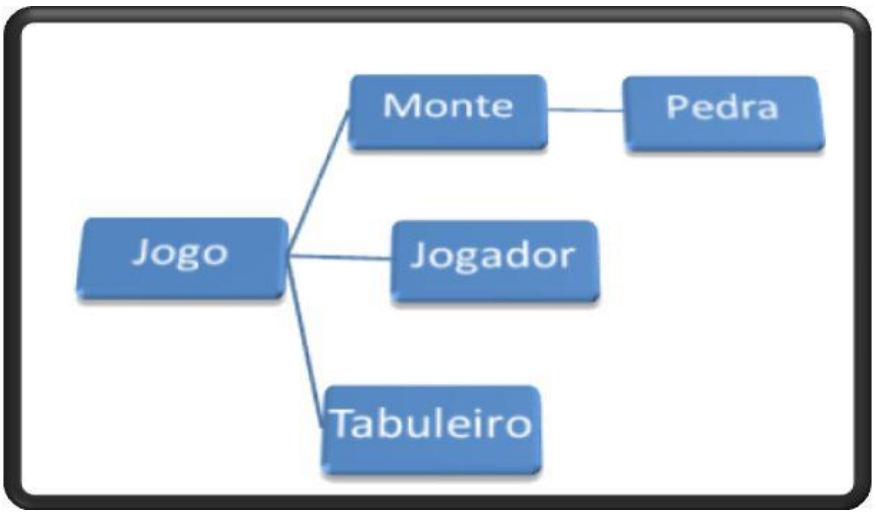
Os jogadores virtuais possuem conhecimento básico, quando for sua vez de jogar ele automaticamente irá procurar uma peça compatível em sua mão. No caso de achar, ele vai jogar a primeira que achar, caso contrário irá buscar no monte, se não existir monte ou o jogador não achar a peça no monte, ele automaticamente vai passar a vez.

Funcionamento do jogo de dominó :

O jogo de dominó funciona da seguinte forma :

Primeiro o jogador físico informa na interface qual será o numero de jogadores virtuais do jogo (1,2,3), para isso ele deve fazer uso da instancia da classe jogo que chama seu método inicia_jogo passando como parâmetro o numero de jogadores informados pelo jogador físico do jogo. No inicio_jogo o método cria o monte e o embaralha e cria o tabuleiro o inicializando e cria os jogadores de acordo com o numero informado e chama a controle de jogo passando como parâmetro o numero de jogadores. A controle de jogo verifica qual o numero de jogadores o jogo deve ter de acordo com o parâmetro e chama o método verificar_maior_peca que verifica nas mãos dos jogadores quem tem a maior pedra. Logo a seguir o jogador com a maior pedra joga tal na mesa e o próximo jogador de acordo com a seqüência decidira de acordo com a maior pedra realiza sua jogada. Assim a cada jogada que é feita tais retornos são feitos para a interface através dos métodos de acesso ao controle do jogo da classe jogo e o jogo acontece. A cada jogada feita pelos jogadores é chamado uma função que verifica se algum deles possui condição de vitoria, onde caso isso aconteça o jogo é finalizado e mostrado o vencedor.
Também pode acontecer a exceção de que todos não tem pedras para jogar no tabuleiro mas ninguém ainda tem condição de vitoria, para isso é chamado um método que contabiliza quantas pedras cada jogador tem e informa quem é o vencedor.

Figura ilustrativa do diagrama de classes do jogo de dominó :



Declaracão das classes do jogo de dominó :

```
class peca
{
	private:
		int direita;
		int esquerda;
	public:
		void criar_peca(int);
		void ler_peca(int cont,int&,int&);
};
class monte
{
	private:
		struct pilha montes;
		peca pecas[28];
	public:
		void criar_monte();
		void ler_peca_monte(int&,int&);
		void embaralhar_monte();
		void retorno_valor_peca(int,int&,int&);
		void ler_topo_monte(int&);
};
class tabuleiro
{
	private:
		int num_jogadas;
		int tabuleiro[60];
		int ponta_D;
		int ponta_E;
	public:
		int inserir(int&,int&,int&);
		void listar_posicao(int&,int&);
		void inicia_tabuleiro();
		void retorno_tabuleiro(monte&,int[],int[],int[]);
};
class jogador
{
	private:
		int mao[21];
		int num_pedras;
	public:
		void criar_jogador(monte&);
		void fazer_primeira_jogada();
		int fazer_jogada(monte&,tabuleiro&);
		int fazer_jogada_fisica(monte&,tabuleiro&,int&,int&);
		void fazer_primeira_jogada(int,int,int,tabuleiro&,jogador&);
		void procurar_peca(int&,int&,monte&,int&,int&);
```

```
                void listar_mao(int&,int);
                int soma_pecas();
                int verificar_vitoria(int&);
                void retorno_mao_virtual(int&);
                void retorno_mao_fisica(monte&,int[],int&);

};
class jogo
{
        private:
                jogador j1,j2,j3,j4;
                monte obj_monte;
                tabuleiro tabua;
                int tabuleiro_D[60],tabuleiro_E[60];
                int tabuleir[60];
                int mao[21];
                int pedra;
        public:
                void inicio_jogo(int);
                void controle_jogo(int);
                void retorno_valor_peca(int,int&,int&);
                int verifica_somatorio_vitoria(int,int&,int,int,int);
                void imprime_tabuleiro(int[]);
                void imprime_mao_fisica(int[],int&);
                void imprime_mao_virtual(int&);
                int imprime_veriricar_somatorio(int&);
                void interface_jogada(int&);
                void retorno_jogada(int&);
                void retorna_maior_pedra(int&);
};
```

Métodos das classes do jogo de dominó

Classe peca

Criada como abstração de uma pedra do jogo, ela é usada para o armazenamento de duas variáveis que se referem aos valores de cada lado da pedra do jogo.

Métodos da classe :

void criar_peca(int);

Objetivos : Criar uma pedra de acordo com o numero do índice passado
Retorno : Não retorna nada
Parâmetros : O índice para criar a pedra

void ler_peca(int cont,int&,int&);

Objetivos : Ler os valores dos lados da pedra de acordo com o valor do índice passado
Retorno : Retorna pelos parâmetros o valor da direita e da esquerda da pedra
Parâmetros : Índice da pedra, Valor da esquerda da pedra,Valor da direita da pedra

Classe monte

Criada como abstração do monte de pedras do jogo, ela é usada para o armazenamento de uma pilha para o controle do monte, e um vetor para o armazenamento de todas pedras do jogo.

Métodos da classe :

void criar_monte();

Objetivos : Criar o monte de pedras
Retorno : não retorna nada
Parâmetros : Não é preciso passar parâmetros para criar o monte

void ler_peca_monte(int&,int&);

Objetivos : Devolver os valores da direita e da esquerda da pedra passada como parâmetro
Retorno : Retorna nas variáveis passadas como parâmetro o valor da esquerda e da direita da pedra
Parâmetros : variável que manda o índice da pedra e que recebe o valor da esquerda e variável que recebe o valor da direita da pedra

void embaralhar_monte();

Objetivos : Embaralha as pecas do monte
Retorno : Não retorna nada
Parâmetros : Não precisa passar parâmetros

void retorno_valor_peca(int,int&,int&);

Objetivos : Retornar o valor das extremidades da pedra de acordo com o índice passado como parâmetro
Retorno : Retorna os valores da pedra de acordo com o índice passado como parâmetro
Parâmetros : Os parâmetros são, o índice da pedra, o retorno da esquerda e o retorno da direita

void ler_topo_monte(int&);

Objetivos : Responsável por retornar o valor da pedra presente no topo do monte
Retorno : Retorna a pedra presente no topo do monte
Parâmetros : O parâmetro é uma variável que recebe o valor da pedra

Classe tabuleiro

Criada como abstração do tabuleiro do jogo, ela é usada para o armazenamento de um vetor que guarda o índice das peças já jogadas, o valor das pontas do jogo as quais podem receber alguma peça e o número de jogadas.

Métodos da classe :

int inserir(int&,int&,int&);

Objetivos : Responsável por inserir uma pedra no tabuleiro passada como parâmetro
Retorno : Não retorna nada

Parâmetros : Os parâmetros são o valor do índice da pedra e seus valores da esquerda e da direita

void listar_posicao(int&,int&);

Objetivos : Responsável por retornar os valores das casas de jogada da esquerda e da direita do tabuleiro
Retorno : Retorna os valores das casas da direita e da esquerda disponíveis para jogar no tabuleiro
Parâmetros : Os parâmetros são o valor da esquerda e o valor da direita do tabuleiro

void inicia_tabuleiro();

Objetivos : Responsável por iniciar valores no tabuleiro
Retorno : Não retorna nada
Parâmetros : Não tem parâmetros

void retorno_tabuleiro(monte&,int[],int[],int[]);

Objetivos : Responsável por retornar a um vetor o tabuleiro
Retorno : Retorna o tabuleiro e as casas da direita e da esquerda de cada pedra
Parâmetros : o monte para obter acesso aos valores de cada pedra, um vetor que deve retornar o tabuleiro e dois outros vetores responsável por retornar os valores da direita e da esquerda de cada posição do tabuleiro

Classe jogador

Criada como abstração do jogador, seja ele físico ou virtual, a classe também é utilizada para o armazenamento do número de peças e de um vetor que age para o armazenamento das peças que o jogador possui.

Métodos da classe :

void criar_jogador(monte&);

Objetivos : Responsável por criar um jogador
Retorno : Não retorna nada
Parâmetros : É necessário passar o monte para ele saber o valor de cada pedra na hora de criar o jogo

int fazer_jogada(monte&,tabuleiro&);

Objetivos : Responsável por fazer as jogadas dos jogadores virtuais
Retorno : Retorna se o jogador fez a jogada ou não
Parâmetros : Para fazer a jogada é necessário passar o monte para buscar os valores das pedras e o tabuleiro para jogar a pedra

int fazer_jogada_fisica(monte&,tabuleiro&,int&,int&);

Objetivos : Responsável por fazer as jogadas do jogador físico
Retorno : Retorna se o jogador fez a jogada ou não

Parâmetros : Passa o valor de monte para buscar os valores das pedras, o tabuleiro para ele fazer a jogada e os valores do índice da pedra na mão
e da opção de jogada do jogador físico(1 - jogar/ 2 - buscar no monte/3 - pular)

void fazer_primeira_jogada(int,int,int,tabuleiro&,jogador&);

Objetivos : Realizar a primeira jogada no tabuleiro
Retorno : Não retorna nada
Parâmetros : Passa como parâmetro os valores do índice da mão do jogador, os valores da esquerda e da direita da pedra e o tabuleiro para jogar a pedra
e o jogador para saber quem exatamente vai jogar(jogador físico ou jogador virtual)

void procurar_peca(int&,int&,monte&,int&,int&);

Objetivos : Responsável por procurar a maior pedra na mão do jogador
Retorno : Retorna o valor da maior pedra encontrada na mão do jogador
Parâmetros : Retorna como parâmetro os valores do somatório, da posição da pedra, e da esquerda e da direita. Recebe como parâmetro o valor do monte

void listar_mao(int&,int);

Objetivos : Responsável por retornar um valor da mão do índice passado como parâmetro
Retorno : Retorna o valor contido na mão sobre o índice passado como parâmetro
Parâmetros : Os parâmetros são o valor da mão retornado como parâmetro e o índice da mão

int soma_pecas();

Objetivos : Responsável por realizar o somatório das pedras da mão do jogador
Retorno : Retorna o valor do somatório
Parâmetros : Não tem parâmetros

int verificar_vitoria(int&);

Objetivos : Responsável por verificar se a mão do jogador não possui nenhuma pedra
Retorno : Retorna como parâmetro se o jogador venceu ou não
Parâmetros : O parâmetro é a variável que recebe o valor que informa se o jogador ganhou ou não

void retorno_mao_virtual(int&);

Objetivos : Retornar o tamanho da mão virtual do jogador
Retorno : Retorna o valor do tamanho da mão virtual do jogador
Parâmetros : O parâmetro é a variável que recebe o valor do numero de pedras da mão do jogador

void retorno_mao_fisica(monte&,int[],int&);

Objetivos : Retornar a mão física do jogador e o numero de pedras presente em tal
Retorno : O retorno é feito nos parâmetros onde a mão é retornada em um vetor e o numero de pedras em uma variável
Parâmetros : Os parâmetros são o monte que serve para verificar os valores das extremidades de cada pedra e o vetor que recebe as pedras da mão e uma variável que recebe a quantidade de pedras da mão do usuário

Classe jogo

Criada como abstração do jogo em si, ela é usada para o armazenamento dos jogadores, do monte de peças e do tabuleiro do jogo.

Métodos da classe :

void inicio_jogo(int);

Objetivos : Responsável por criar os jogadores de acordo com o numero de jogadores informados por parâmetro
Retorno : Não retorna nada
Parâmetros : O parâmetro é o numero de jogadores que é passado a ela criar

void controle_jogo(int);

Objetivos : Responsável por controlar as jogadas do jogo de acordo com o primeiro jogador a jogar
Retorno : Não retorna nada
Parâmetros : O parâmetro é o numero de jogador que indica quantos jogadores o controle deve controlar

void retorno_valor_peca(int,int&,int&);

Objetivos : Responsável por retornar o valor de uma pedra de acordo com o índice informado
Retorno : Retorna nos parâmetros os valores da esquerda e da direita da pedra
Parâmetros : Os parâmetros são o índice da pedra e os valores da esquerda e da direita da pedra

int verifica_somatorio_vitoria(int,int&,int,int,int);

Objetivos : Retornar 1 se o retorno das jogadas dos jogadores é igual a 2
Retorno : Retorna 1 se o retorno dos jogadores for igual a 2 e retorna no segundo parâmetro o valor do jogador que venceu
Parâmetros : Os paramentos são o numero de jogadores, o retorno do jogador 1, o retorno do jogador2, o retorno do jogador 3 e o retorno do jogador 4

void imprime_tabuleiro(int[]);

Objetivos : Retornar o tabuleiro para a interface
Retorno : Retorna em um vetor de inteiros passados como parâmetros o tabuleiro

Parâmetros : O parâmetro é um vetor de inteiros que recebe o valor do tabuleiro

void imprime_mao_fisica(int[],int&);

Objetivos : Retornar para a interface o numero de pedras e a mão do jogador físico
Retorno : Retorna os valores da mão e o numero de pedras
Parâmetros : Os parâmetros são o vetor que recebe a mão e uma variável inteira que recebe o numero de pedras

void imprime_mao_virtual(int&);

Objetivos : Retornar para a interface o numero de pedras da mão do jogador virtual
Retorno : Retorna o numero de pedras da mão do jogador virtual
Parâmetros : O parâmetro é um inteiro que recebe o valor do numero de pedras da mão virtual

int imprime_veriricar_somatorio(int&);

Objetivos : Retornar o valor do somatório das pedras da mão do jogador
Retorno : Retorna o somatório
Parâmetros : Recebe um inteiro que é retornado

void interface_jogada(int&);

Objetivos : Entrar com o valor da jogada do usuário na interface para o método de controle de jogo

Retorno : Não retorna valores
Parâmetros : O parâmetro é a variável que carrega o valor para o método utilizar

void retorno_jogada(int&);

Objetivos : Retornar o resultado da jogada do jogador físico
Retorno : Retorna o valor da jogada de acordo com o tratamento de erros da fazer jogada física
Parâmetros : O parâmetro é um inteiro que retorna o valor par a chamada da interface

void retorna_maior_pedra(int&);

Objetivos : retornar o numero do jogador que tem a maior pedra para a interface
Retorno : Retorna no parâmetro o valor da maior pedra
Parâmetros : O parâmetro é responsável por retornar o numero do jogador com a maior pedra