Diagrama

Descrição gerada automaticamente com confiança baixa

**Faculdade de Ciências e Tecnologia**

**Licenciatura em Engenharia**

**Eletrotécnica e de Computadores**

**Estruturas de Dados e Algoritmos**

**Mini Projeto I**

**Jogo da Forca**

Trabalho realizador por:

Eduardo Rocha Falvo

(2021192252)

Gabriel Santos Corrêa da Silva

(2021104069)

PL6

25 de novembro de 2022

Sumário

**1 Introdução .......................................................................................... 3**

**2 Diagrama Estrutural do Código .......................................................... 3**

**3 Estrutura Geral .................................................................................. 3**

3.1 Menu ............................................................................................................ 4

3.2 Novo Jogo ..................................................................................................... 4

3.3 Carrega Jogo ................................................................................................. 4

3.4 Modo de Jogo ............................................................................................... 5

3.5 Histórico ....................................................................................................... 5

3.6 Jogo .............................................................................................................. 5

3.7 Avança Jogo ................................................................................................. 6

3.8 Imprime Tempo ........................................................................................... 6

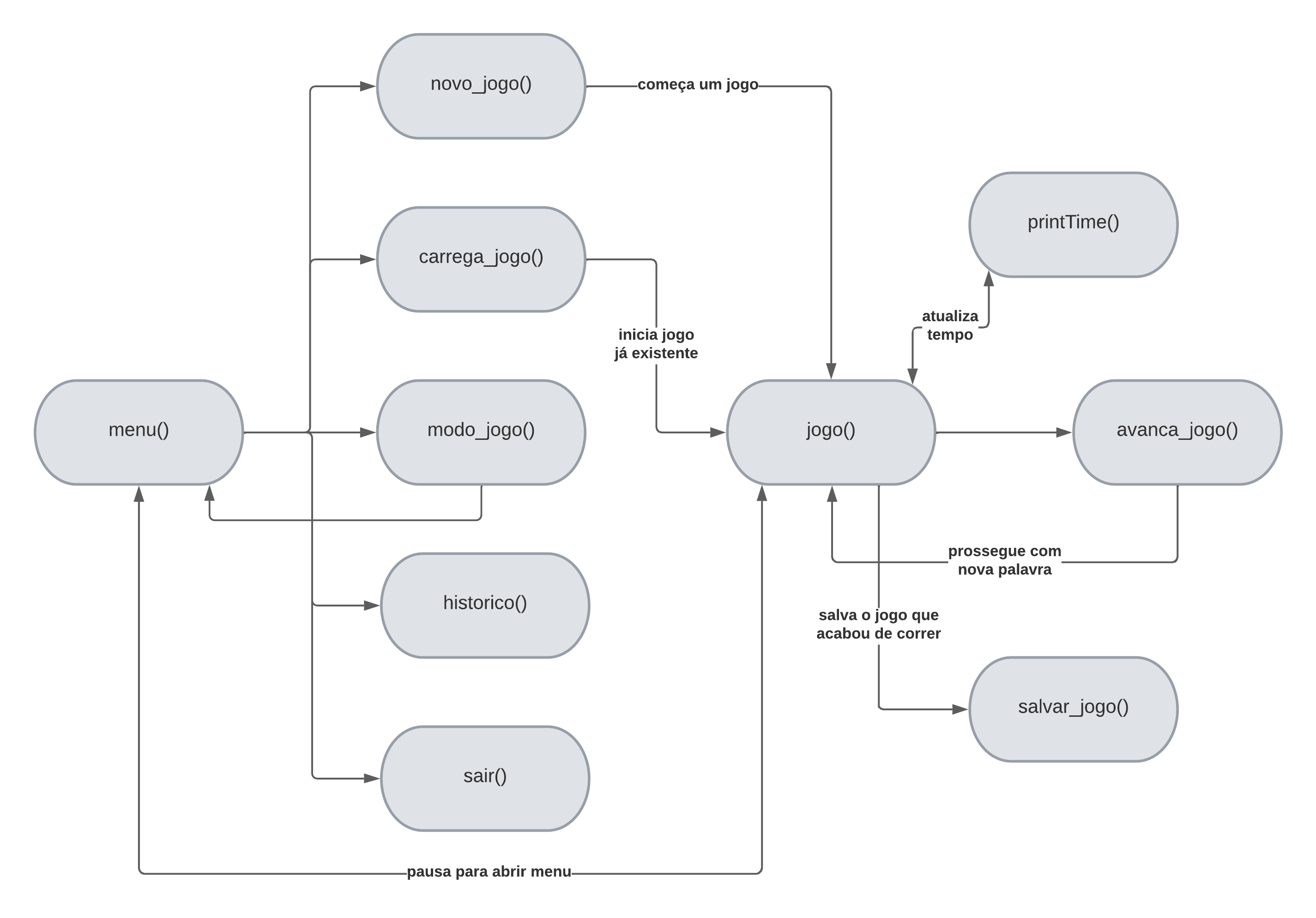
3.9 Sair ............................................................................................................... 6

**4 Conclusão ........................................................................................... 7**

**1. Introdução**

O Jogo da Forca (HangMan) é um jogo em que o jogador tem que acertar uma palavra, tendo como dica o número de letras e o tema ligado à palavra. Neste contexto, foi-nos proposto como MiniProjeto I da cadeira de Estruturas de Dados e Algoritmos que, utilizando a linguagem de programação C++ e suas bibliotecas disponíveis, criássemos nosso próprio jogo a partir de especificações predefinidas pelos professores da cadeira. Por fim, vamos abordar neste relatório as metodologias e lógicas utilizadas pelos integrantes do grupo ao desenvolver e executar o algoritmo do jogo.

**2. Diagrama Estrutural do Código**

****

**3. Estrutura Geral**

Ao analisar a lógica do Jogo da Forca e verificar as especificações estabelecidas pelos Professores; concluímos que a melhor estrutura a ser usada no desenvolvimento do programa seria a lógica da Programação Orientada a Objetos com o uso de classes e seus respectivos membros (atributos e métodos).

Começamos por implementar a nossa classe Jogo, que consta com atributos que fazem parte de um jogo da forca, como: nome do jogador, a palavra a ser descoberta, o tamanho desta palavra, o tema referente à palavra, tentativas que já foram feitas, acertos já feitos, os pontos feitos pelo jogador, o modo de jogo (elementar, básico ou médio – relativo aos temas e aleatoriedade de palavras), a dificuldade do jogo (fácil, médio ou difícil – relativo ao tempo) e o tempo que o jogador demorou. Além disso, temos alguns outros atributos para auxiliar no progresso do nosso jogo na parte algorítmica, como: uma tabela com as letras já acertadas da palavra, uma tabela com as palavras já acertadas (para não haver repetição destas no modo aleatório) e uma tabela com as letras erradas (para não deixar o usuário fazer os mesmos erros e mostrar a ele quais letras já foram tentadas).

E, em adição aos atributos, que são inicializados todos vazios por um construtor por omissão, Jogo(), e são atualizados ao decorrer do jogo, temos os métodos relacionados a nossa classe, que, além de expressos didaticamente em formato de diagrama, definem a ordem de funcionamento do jogo e que serão mais bem explicados a seguir.

Por fim, vale mencionar que criamos ainda três estruturas, todas relacionadas com a biblioteca <sys/time.h> e implementadas para a criação do temporizador do jogo, componente que será descrito adiante, e que utilizamos a forma de escrita e leitura em ficheiros binários para fazer uma base de dados das palavras relacionadas a três temas e jogos e pontuações passada.

**3.1 Menu**

No método menu(), inicialmente limpamos o terminal com o comando **system**("CLS"), ferramenta utilizada sistematicamente em nosso código para deixar o jogo o mais profissional possível. Após isso, a partir da biblioteca <iomanip> , imprimimos a parte visual do menu de modo formatado e aguardamos que o usuário selecione a opção desejada: iniciar um novo jogo, carregar um jogo preexistente, selecionar o modo de jogo, ver o histórico dos jogadores, ou sair do jogo.

Assim, dependendo da opção selecionada pelo utilizador, seguimos para um dos outros cinco métodos criados.

**3.2 Novo Jogo**

Em novo\_jogo(), criamos o método que irá inicializar os atributos da classe todos vazios e que fará os preparativos para inicializarmos um jogo do zero. O programa, após zerar as variáveis da classe, começa por verificar o modo de jogo atual da aplicação. No caso de o modo ser básico ou médio, pede-se ao usuário que este escolha um dos três temas disponíveis, além da dificuldade do jogo e o nome deste jogador.

Feito isso, primeiramente atribuímos ao atributo “dificuldade” um valor relativo ao nível escolhido e que é utilizado para definir, mais tarde, o tempo que o usuário terá para acertar a palavra; se for a dificuldade “FÁCIL”, o valor atribuído é maior, para que o jogador tenha mais tempo e seja mais simples de alcançar a vitória; caso seja a dificuldade “DIFÍCIL”, é o oposto.

Após verificar a dificuldade, acessamos a base de dados em ficheiro binário feita e selecionamos a palavra aleatoriamente a partir da função dentro do escopo do tema escolhido. Atribuímos ao atributo “palavra” a selecionada e ao atributo “tamanho\_palavra” a quantidade de caracteres desta.

**3.3 Carrega Jogo**

**3.4 Modo de Jogo**

Neste curto método, modo\_jogo(), pedimos ao utilizador da aplicação que este escolha entre os tipos de jogo (elementar, básico e médio) e armazena sua decisão no atributo “escolha”.

**3.5 Histórico**

**3.6 Jogo**

Sendo um dos principais e mais complexos métodos do nosso programa, jogo() inicia-se pegando o tempo atual a partir da biblioteca <sys/time.h> e armazenando-o na estrutura “before”. Após isso, entramos em um ciclo que permanece rodando até que o número de tentativas feitas pelo usuário se iguale a oito, ou seja, cometa oito erros.

Enquanto a condição de erros não seja satisfeita, o usuário não acerte todas as letras ou ainda o tempo não acabe, o programa continua pedindo a este um novo input de um char e comparando com as letras da palavra. Se a letra enviada já estiver sido usada o jogador é avisado e pede-se que este insira outra letra, caso a letra ainda não tenha sido inserida, o programa verifica se esta letra pertence a palavra a partir da função , que retorna a posição do caractere na string e atualiza o espaço “\_” respectivo a letra encontrada; além de incrementar o atributo acertos.

Texto

Descrição gerada automaticamente

No caso do número de letras acertadas seja igual ao tamanho da palavra selecionada aleatoriamente, temos que o jogador acertou a palavra, logo seus pontos são contabilizados da devida forma (oito menos a quantidade de erros para o modo elementar e o tempo restante para os modos básico e médio) e avançamos para outro jogo com a chamada do método “avança\_jogo()”, que será detalhado adiante.

É ainda importante ressaltar que após cada input de caracteres, guardamos na terceira estrutura, “after”, o horário atual e atualizamos o atributo tempo e o nosso contador decrescente na tela.

**3.7 Avança Jogo**

Este método, avança\_jogo(), é responsável por dar continuidade ao jogo após o jogador acertar uma palavra, de modo a reiniciar os acertos, letras certas e erros, armazenar a palavra que já foi acertada no jogo anterior no atributo “palavrasJogadas” e abrir o ficheiro de palavras em modo leitura para selecionar outra palavra, não repetida, do mesmo tema para dar continuidade ao jogo.

**3.8 Imprime Tempo**

Apesar de curto, é um importante método da nossa classe, por ser responsável por fazer a conversão dos valores de tempo armazenados nas estruturas criadas e retornar um valor real com a diferença entre o tempo atual e o inicial, de modo a mostrar ao usuário quanto tempo este ainda tem para acerta a palavra.

**3.9 Sair**

Sendo o último método da nossa classe, sair() é chamado quando o usuário encontra-se no menu() e deseja fechar a aplicação. Antes de finalizar o programa, é verificado se há algum jogo em curso no momento através do atributo “emJogo” e, caso a resposta seja “true”, avisa o jogador que há um jogo em curso e pergunta a este se há a necessidade de salvar o jogo antes de sair. No caso de a resposta do jogador ser sim, é chamado o método salvarjogo() e depois é finalizado o programa.

**4. Conclusão**

Por fim, concluímos que as tecnologias usadas por nós, como a Orientação a Objetos a partir da criação da classe Jogo, combinado com a manipulação de ficheiros binários, tanto em escrita da data base de palavras e jogos passados, quanto na leitura dessas informações, e associado à cronometragem do tempo com a biblioteca exterior supracitada, foi a mais eficiente possível de modo a resultar no projeto apresentado.