

Técnicas de Programação - Projeto III - 1o Informática Matutino - 2021

Este projeto tem por objetivo o desenvolvimento das habilidades do aluno na criação de programas por meio da Programação Visual, do Pensamento Orientado a Objetos, do manuseio de arquivos textos e do tratamento de eventos. Como tema, usaremos o Jogo da Forca.

O programa deverá abrir um arquivo texto contendo, em cada linha, uma string com uma palavra ocupando 15 caracteres e, a partir do 16o caracter, uma string contendo uma frase que forneça uma dica da palavra. Esse arquivo texto poderá ter ao redor de 100 linhas. Por exemplo:

IPTU	Imposto sobre imóveis urbanos
BALEIA	Cetáceo
SHAKESPEARE	Escritor inglês
FERRAMENTA	Dispositivo que forneça vantagem mecânica para realizar tarefas

Classe Dicionario

Codifique uma classe **Dicionario** que permita armazenar, como atributos, a palavra e sua dica, propriedades de acesso a elas e mapeie os campos do arquivo usando constantes de tamanho e início dos campos. Nessa classe, crie um método para ler uma linha de dados e armazenar essa linha nas propriedades Palavra e Dica e um método que retorne os dados dos atributos, formatados de acordo com a estrutura do arquivo. A classe **Dicionario** será muito semelhante, em termos de atributos, propriedades e métodos, à classe **Funcionario** que estudamos em aula, mas adaptada ao armazenamento e tratamento de dados de dicionário (palavra e dica).

Essa classe deverá também ter um atributo `bool[] acertou`, que é um vetor de valores lógicos que sempre será iniciado com 15 posições valendo **false**. Esse atributo será usado para sabermos se a letra de uma posição *i* qualquer da palavra foi acertada pelo jogador no jogo de Forca. Explicaremos mais sobre isso depois.

Nessa classe, codifique as propriedades e métodos necessários para:

- Retornar true se a letra sorteada está na palavra (pode haver mais de uma posição) atualizando o vetor `acertou` com true nas posições onde a encontrou; retornar false caso a letra sorteada não exista na palavra
- Retornar o vetor `acertou`, contendo true nas posições em que letras indicadas pelo jogador foram encontradas na palavra e false nas posições em que as letras correspondentes ainda não foram indicadas.

Classe VetorDicionario

Crie essa classe usando como base a classe `VetorFuncionario`, que estudamos em aula. A classe `VetorDicionario` terá o vetor `dados`, que permitirá armazenar objetos da classe `Dicionario`. Ela terá os mesmos atributos, propriedades e métodos da classe `VetorFuncionario`, mas adaptados para armazenamento e acesso de dados da classe `Dicionario`. Por exemplo, se `posicaoAtual = 2`, acessaremos a terceira posição desse Vetor:

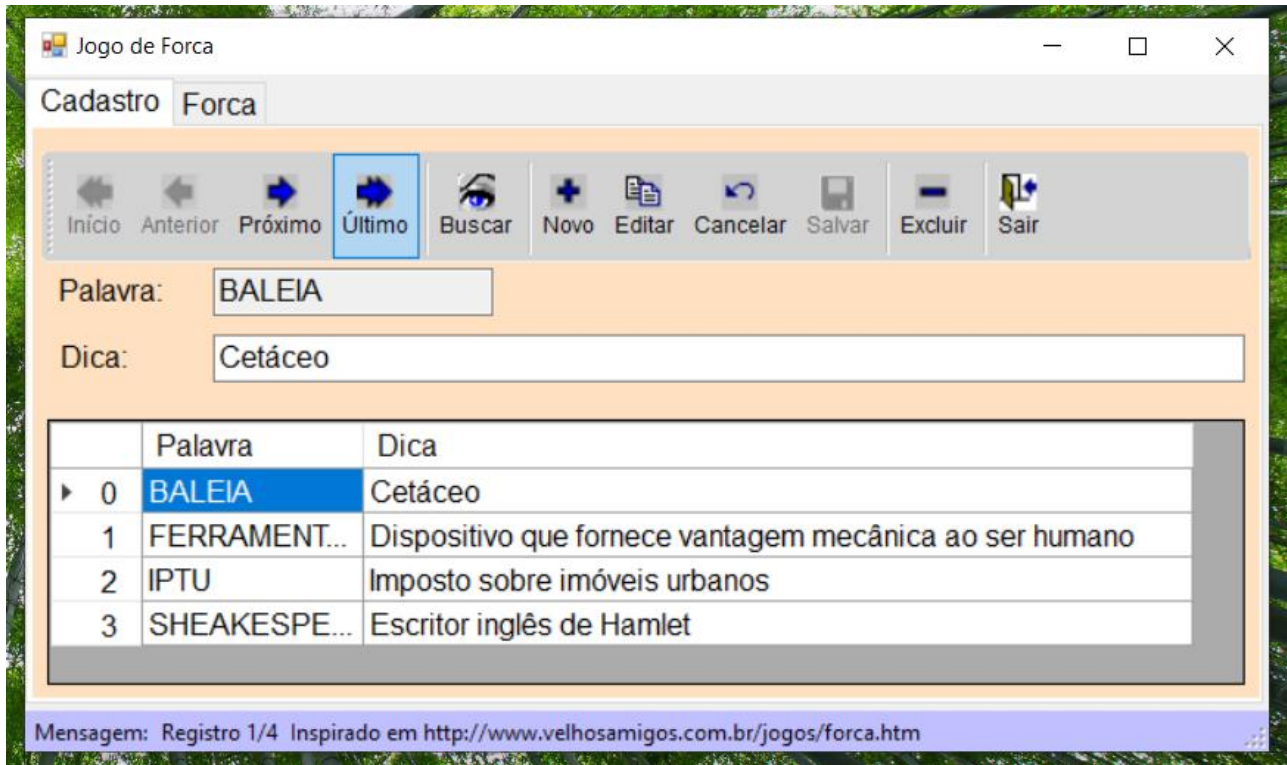
VetorDicionario		dados		qtosDados: 4						
		palavra	dica	acertou						
0		BALEIA	Cetáceo	0	1	2	...	13	14	
							...			
1		FERRAMENTA	Dispositivo que forneça vantagem...	0	1	2	...	13	14	
							...			
posicaoAtual:→ 2		IPTU	Imposto sobre imóveis urbanos	0	1	2	...	13	14	
							...			
3		SHAKESPEARE	Escritor inglês	0	1	2	...	13	14	
							...			
...										

O programa de aplicação deverá ter um `TabControl` com duas guias: uma guia de **Cadastro** de palavras e dicas (ou seja, cadastro do dicionário) e outra guia para o **jogo** de Forca.

No início da execução do programa, o nome do arquivo no disco com os dados deverá ser solicitado através de um `OpenFileDialog`. Após isso, o arquivo selecionado deverá ser aberto e

lido, linha a linha, e seus dois campos armazenados em objeto da classe Dicionario que será incluído ao final do vetor interno dados de um objeto da classe VetorDicionario.

Guia Cadastro



Nessa aba, temos um ToolStrip com vários botões, campos para digitação e exibição de dados do arquivo aberto e um DataGridView para exibição dos dados que foram armazenados no VetorDicionario. Ao entrarmos nessa guia, os dados presentes no VetorDicionario devem ser percorridos e apresentados, linha a linha, no DataGridView, sendo que suas colunas serão (posição, palavra e dica).

Da mesma forma que a classe VetorFuncionario em que se baseia, a classe VetorDicionario deverá ter o atributo **PosicaoAtual** que indicará a posição, no VetorDicionario da palavra e sua dica atualmente exibidas nessa tela. Sempre que se pressionar um botão de navegação (Início, Voltar, Avançar e Último), deve-se mudar o valor dessa variável de forma consistente, para que ela indique a posição da nova palavra e dica para as quais se navegou. Em seguida, a palavra e dica recém indexadas no VetorDicionario devem ser exibidas nos Textbox edPalavra e edDica, respectivamente. Ou seja, essa guia seguirá os mesmos padrões que estudamos durante o desenvolvimento do aplicativo de manutenção de funcionários.

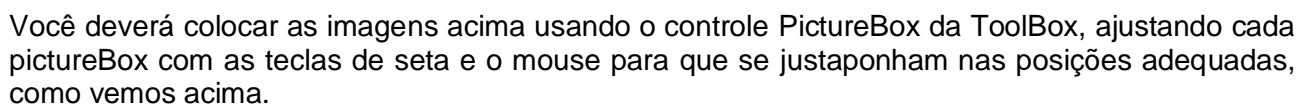
Quando se pressionar o botão [Incluir], deve-se indicar que o programa está em **Modo de Inclusão**, limpar os Textbox, colocar o cursor em edPalavra e, após o usuário digitar uma nova palavra e tirar o cursor desse Textbox (evento Leave), procurar a palavra digitada no vetor dados interno de VetorDicionario, usando o método de pesquisa binária Existe dessa classe. Se já existir, avisar o usuário e cancelar a operação, exibindo os dados indicados por **PosicaoAtual**. Se não existir, colocar o cursor em edDica e permitir a digitação da dica.







Quando o usuário pressionar o botão [Salvar], verificar se o programa está em **Modo de Inclusão**. Se estiver, inclua a nova palavra e a nova dica na posição que mantenha em ordem o vetor dados interno de VetorDicionario. Modifique **PosicaoAtual** para que essa variável indique a posição onde o objeto com os novos dados (palavra, dica) foi incluído no VetorDicionario. Ou seja, o fluxo de operação é o mesmo que fizemos no processo de inclusão de um Funcionario, mas adaptado aos campos de Dicionario.



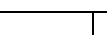
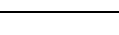
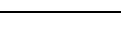



Ao pressionar [Alterar], colocar o programa em **Modo de Edição**. Colocar o cursor em edDica e permitir que o usuário mude a dica. Quando for pressionado o botão [Salvar], verificar se está em Modo de Edição e, se estiver, substituir pelo novo valor a dica armazenada na posição **PosicaoAtual** do vetor dados interno de VetorDicionario.

Quando o botão [Excluir] for pressionado, deve-se remover do VetorDicionario o objeto armazenado na posicao **PosicaoAtual**, depois de confirmação pelo usuário.



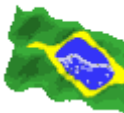


Guia Forca



 <p>Forca_01</p> <p>Forca_02</p>	 <p>Forca_03</p>	 <p>Forca_05</p>	 <p>Forca_08</p>	 <p>Forca_09</p>	 <p>Forca_13</p>
---	---	---	---	---	---

							
Forca_05	Forca_09	Forca_07	Forca_10	Forca_14	Forca_16	Forca_17	Forca_105

Há, ainda as figuras adicionais:

 Forca_2_07	 Forca_2_04	 Forca_2_03	 enforcado	 Forca_08
--	---	---	--	--

Nas PictureBox com as imagens que formam o personagem, configure a propriedade Visible com False. Alterne essa propriedade para true quando quiser que uma das imagens apareça, significando que a pessoa teve mais uma parte exibida ao ser cometido um erro de letra.

Funcionamento geral do jogo

Quando o jogador pressionar o botão [Iniciar], deverá ser feito um sorteio do número do elemento que será acessado no vetor interno **dados** do VetorDicionario, através do uso da função Random.Next(VetorDados.Tamanho), que retorna um valor inteiro entre 0 e o tamanho do vetor interno **dados** menos 1. Você pode, também, ocultar a guia Cadastro nesse momento, enquanto houver jogo sendo realizado. Assim, evitará que o jogador veja o dicionário e trapaceie.

Se o checkBox [Com Dica] estiver selecionado, deve-se exibir a dica acessada lida no label [Dica:].

Deve-se então limpar o DataGridView abaixo dos botões, configurando-o para que o seu número de colunas seja igual ao número de caracteres (sem os espaços à direita) da palavra lida do arquivo. Pesquise a classe String para saber como tornar uma palavra em Maiúsculo e remover os espaços à direita de **palavra**. Isso deve ser feito para comparar adequadamente as letras com os textos dos botões que foram pressionados.

O evento Click dos botões de cada uma das letras terá comportamento praticamente igual, variando apenas a letra que corresponde ao Text do botão. Você pode, portanto, codificar um único tratador de evento Click e associá-lo a todos os botões. Usando (sender as Button).Text você acessa a letra que está definida no Text de cada botão, pois o parâmetro sender representa o botão que foi clicado.

Após isso, quando o usuário clicar num dos botões de letras, deve-se verificar se essa letra existe na palavra. Se existir, deve-se exibi-la na posição do DataGridView correspondente ao local onde essa letra ocorre na palavra e marcar com true essa mesma posição no vetor **acertou** do objeto Dicionario que foi sorteado e está armazenado no vetor interno **dados** de VetorDicionario.

Se a letra não ocorrer, deve-se tornar visível na forca a figura que corresponde à parte da "pessoa enforcada" correspondente ao número do erro cometido (até 8 erros).

Se o usuário selecionar a [Com dica], você deve estabelecer um tempo máximo para ele acertar a palavra. Use o timer tmrTempo para controlar esse tempo, de um em um segundo, diminuindo-o até chegar a zero. Nesse caso, se o usuário ainda não tiver acertado a palavra, ele perde o jogo.

Quando o jogador acertar uma letra, seu número de pontos deve ser aumentado. Sempre que o jogador acertar uma letra, ele ganha 1 ponto. Ao errar, perde um ponto. Se acertar a palavra, ele ganha o jogo e isso deve ser anunciado.

Se o jogador perder, a sua imagem ascendendo aos céus deve ser mostrada, de preferência em uma animação mostrando essa ascensão (você pode usar um timer para controlar isso).

Integração com Arduino

Se o usuário marcar o checkBox Arduino, tornar visível o label e o textbox referentes à porta serial que será usada para a integração deste programa com o kit arduino ou seu simulador. Sempre que houver um erro ou as demais condições estipuladas na parte do projeto referente ao Arduino e esse checkBox estiver marcado, faça a transferência das informações necessárias para que o número de erros represente a quantidade de leds que serão acessos, ou uma letra para indicar vitória ou uma letra para indicar derrota, para que o sensor Arduino colete esse caracter e o programa que executa o trate de acordo com o especificado.

Orientações Adicionais

Os nomes das variáveis, os comentários adequados e o alinhamento dos comandos serão fator de avaliação.

O trabalho deve ser desenvolvido em dupla, e deve ser entregue um relatório técnico sobre o desenvolvimento do projeto, com as seguintes partes:

Identificação dos autores (nomes e RAs) e do trabalho

Introdução

Descrever objetivos do trabalho

Desenvolvimento

- Descrever datas, horas e tempo de desenvolvimento
- Descrever erros encontrados e soluções aplicadas
- Descrever dificuldades encontradas
- Descrever auxílio da monitoria
- Fontes de pesquisa

Conclusão

Descrever o que foi aprendido e as conclusões a que chegou.

IMPORTANTE

- Informe em seu relatório as fontes de pesquisa que utilizou e o assunto pesquisado. Para isso, sempre anote essas informações em um caderno ou arquivo de trabalho, para não perdê-las.
- Trabalho desenvolvido no Visual Studio em C#.
- Comentar corretamente o programa e o código programado
- No início dos arquivos de código, digitar comentário com o RA e nome do(a) aluno(a)
- Relatório de desenvolvimento** deve ser feito num arquivo cujo nome é:
 < RA_Menor><RA_Maior>RelatorioProj3TP.pdf
- Deve ser escrito em Word ou outro editor de texto e salvo no formato PDF para entrega
- Material a ser entregue: pasta completa, com nome raMenor_raMaior_Proj3, contendo todo o projeto desenvolvido, arquivos fonte, formulários, classes, arquivos de dados e o relatório. A pasta deve compactada antes de entregue e o nome do arquivo compactado deve ser raMenor_raMaior_Proj3.rar.
- A entrega será feita no Classroom, na atividade do Projeto 3.