

Deepseek (Perguntas e Respostas)

PEDIDO :

faça um jogo de perguntas e respostas no flowgorithm .

RESPOSTA DO DEEPSEEK :

Claro! Vou te guiar na criação de um jogo de perguntas e respostas simples usando o Flowgorithm. O Flowgorithm é uma ferramenta visual para criar algoritmos e fluxogramas. Vamos criar um jogo básico com três perguntas e respostas.

Passos para criar o jogo no Flowgorithm:

1. ****Início do Programa:****

- Abra o Flowgorithm e crie um novo arquivo.
- Adicione um símbolo de "Início" (Start) ao fluxograma.

2. ****Declaração de Variáveis:****

- Adicione um símbolo de "Declaração" (Declare) para criar as variáveis que serão usadas no jogo.

- Crie variáveis para armazenar as respostas do usuário e a pontuação. Por exemplo:

- `resposta1`, `resposta2`, `resposta3` (para armazenar as respostas do usuário).

- `pontuacao` (para armazenar a pontuação do usuário).

3. ****Primeira Pergunta:****

- Adicione um símbolo de "Saída" (Output) para exibir a primeira pergunta. Por exemplo:

- "Qual é a capital da França?"

- Adicione um símbolo de "Entrada" (Input) para capturar a resposta do usuário e armazená-la na variável `resposta1`.

4. ****Verificação da Resposta:****

- Adicione um símbolo de "Decisão" (Decision) para verificar se a resposta está correta.

- Na condição, compare `resposta1` com a resposta correta ("Paris").

- Se a resposta estiver correta, incremente a variável `pontuacao` em 1.

- Adicione símbolos de "Saída" (Output) para informar ao usuário se ele acertou ou errou.

5. ****Segunda Pergunta:****

- Repita os passos 3 e 4 para a segunda pergunta. Por exemplo:

- "Quantos planetas existem no sistema solar?" (Resposta correta: "8").

- Armazene a resposta do usuário em `resposta2` e verifique se está correta.

6. ****Terceira Pergunta:****

- Repita os passos 3 e 4 para a terceira pergunta. Por exemplo:

- "Quem escreveu 'Dom Quixote'?" (Resposta correta: "Miguel de Cervantes").

- Armazene a resposta do usuário em `resposta3` e verifique se está correta.

7. ****Exibição da Pontuação Final:****

- Adicione um símbolo de "Saída" (Output) para exibir a pontuação final do usuário. Por exemplo:

- "Sua pontuação final é: " + `pontuacao`.

8. ****Fim do Programa:****

- Adicione um símbolo de "Fim" (End) para encerrar o programa.

Exemplo de Fluxograma:

1. ****Início****

2. ****Declare**** `resposta1`, `resposta2`, `resposta3`, `pontuacao`

3. ****Output**** "Qual é a capital da França?"

4. ****Input**** `resposta1`

5. ****Decision**** `resposta1 == "Paris"`

- Se verdadeiro: ****Output**** "Correto!" e

****Increment**** `pontuacao` em 1.

- Se falso: ****Output**** "Errado!"

6. ****Output**** "Quantos planetas existem no sistema solar?"

7. ****Input**** `resposta2`

8. ****Decision**** `resposta2 == "8"`

- Se verdadeiro: ****Output**** "Correto!" e

****Increment**** `pontuacao` em 1.

- Se falso: ****Output**** "Errado!"

9. ****Output**** "Quem escreveu 'Dom Quixote'?"

10. ****Input**** `resposta3`

11. ****Decision**** `resposta3 == "Miguel de Cervantes"`

- Se verdadeiro: ****Output**** "Correto!" e

****Increment**** `pontuacao` em 1.

- Se falso: ****Output**** "Errado!"

12. ****Output**** "Sua pontuação final é: " + `pontuacao`

13. ****Fim****

Dicas:

- Você pode adicionar mais perguntas seguindo o mesmo padrão.

- Para tornar o jogo mais interessante, você pode adicionar um limite de tempo ou dicas para cada pergunta.

Agora você pode executar o fluxograma no Flowgorithm e testar o jogo de perguntas e respostas!

Deepseek (COBRA NA CAIXA !)

PEDIDO :

- A história do jogo se passa dentro de uma tumba egípcia onde dois exploradores ficaram presos. No centro da sala, há duas caixas: uma delas contém o botão para abrir a porta, enquanto uma outra esconde uma cobra mortal. A cada rodada, o local do botão e da cobra muda de caixa de forma aleat é ria, aumentando o desafio e a imprevisibilidade do jogo.
 - Cada jogador deve escolher um nome a partir de uma lista de sete opções predefinidas, e o computador sorteará quem começa;
 - O jogador escolhido seleciona uma caixa, e o sistema informará se há uma cobra, o botão ou se está vazio;
 - O jogo prosseguirá até que alguém encontre a cobra (perda) ou o botão (vence).
 - Ao término, o jogador poderá optar por jogar novamente ou retornar ao menu principal.
- faça esse jogo em flowgorithm com esses passos
- ## ATUALIZAÇÃO DO PEDIDO :
- faça em .jpg
- ## RESPOSTA DO DEEP SEEK :
- infelizmente, não posso criar ou enviar arquivos **“.jpg”** (formato do Flowgorithm) diretamente aqui, mas posso te ajudar a **“construir o jogo passo a passo”** no Flowgorithm e fornecer o **“código equivalente em pseudocódigo”** para você replicar.
- Vamos dividir em **“etapas”**:
-
- ### “1. Estrutura Básica do Flowgorithm (.jpg)”**
- Você pode criar um novo arquivo no Flowgorithm e seguir estes passos:
- ### “Variáveis Globais”**
- **‘jogadores’ (Array de String)** → Lista de nomes pré-definidos.
 - **‘jogador1’, ‘jogador2’ (String)** → Nomes escolhidos.
 - **‘jogadorAtual’ (String)** → Quem está jogando.
 - **‘caixas’ (Array de Inteiro)** → Tamanho 5 (índices 1 a 5).
 - **‘rodada’ (Inteiro).**
 - **‘jogarNovamente’ (Boleano).**
-
- ### “2. Pseudocódigo para o Fluxo Principal”**
- Aqui está o **“pseudocódigo”** que você pode traduzir para o Flowgorithm:
- ```
“plaintext
INÍCIO

 // Menu Principal

 Exibir "Bem-vindo à Tumba Egípcia"

 Exibir "1. Jogar"

 Exibir "2. Sair"

 Ler opção

 Se opção = 1 Então

 // Configuração do Jogo

 jogadores = ["Amun", "Hefertar", "Tutankhamon", "Cleópatra", "Ramsés", "Anúbis", "Horus"]

 Exibir "Escolha seu nome (1 a 7):"

 Para i = 1 até 7 Faça

 Exibir i & ". " & jogadores[i]

 Fim Para

 Ler escolha1

 jogador1 = jogadores[escolha1]

 Exibir "Segundo jogador, escolha seu nome (1 a 7):"

 Ler escolha2

 jogador2 = jogadores[escolha2]

 // Sorteia quem começa

 jogadorAtual = EscolherAleatorio(jogador1, jogador2)

 Exibir jogadorAtual & " começa!"

 // Loop do Jogo

 jogarNovamente = Verdadeiro

 Enquanto jogarNovamente = Verdadeiro Faça

 // Reinicia as caixas

 botao_pos = Aleatorio(1, 5)

 cobra_pos = Aleatorio(1, 5)

 Enquanto cobra_pos = botao_pos Faça

 cobra_pos = Aleatorio(1, 5)

 Fim Enquanto

 Para i = 1 até 5 Faça

 Se i = botao_pos Então

 caixas[i] = 1

 Senão Se i = cobra_pos Então

 caixas[i] = -1

 Senão

 caixas[i] = 0

 Fim Se

 // Rodada

 Exibir "Escolha uma caixa (1 a 5):"

 Ler escolha

 Se caixas[escolha] = 1 Então

 Exibir "Porta aberta! " & jogadorAtual & " venceu!"

 jogarNovamente = Falso

 Senão Se caixas[escolha] = -1 Então

 Exibir "Cobra mortal! " & jogadorAtual & " perdeu!"

 jogarNovamente = Falso

 Senão

 Exibir "Caixa vazia. Próximo jogador!"

 Se jogadorAtual = jogador1 Então

 jogadorAtual = jogador2

 Senão

 jogadorAtual = jogador1

 Fim Se

 Fim Enquanto

 // Pergunta se quer jogar novamente

 Exibir "Jogar novamente? (S/N)"

 Ler resposta

 Se resposta = "N" ou resposta = "n" Então

 jogarNovamente = Falso

 Fim Se

 Fim Enquanto

 Senão

 Exibir "Até mais!"

 Fim Se

FIM
"
```
- 
- ### “3. Como Implementar no Flowgorithm”**
1. **“Crie os símbolos”** no Flowgorithm:
- **“Input/Output”** para interações com o usuário.
  - **“Processar”** para atribuições e operações.
  - **“Condicionais (If/Else)”** para verificar caixas e vitória/derrota.
  - **“Loop (While/For)”** para repetir rodadas.
2. **“Funções Úteis”**:
- **‘Aleatorio(mín, max)’** → Gerar posições do botão e da cobra.
  - **‘EscolherAleatorio(a, b)’** → Decide quem começa.
3. **“Exemplo Visual”**:
- Veja como ficaria parte do fluxograma:
- ```
graph TD
    INICIO --> Menu[Menu Principal 1-Jogar / 2-Sair]
    Menu --> EscolhaNomes[Escolha de Nomes jogador1, jogador2]
    EscolhaNomes --> SorteiaAtual[Sorteia jogadorAtual]
    SorteiaAtual --> Loop[Loop: Reinicia caixas (botão e cobra em posições aleatórias)]
    Loop --> JogadorEscolhe[Jogador escolhe uma caixa]
    JogadorEscolhe --> EBotao{É botão? -> Vitória}
    EBotao --> ECobra{É cobra? -> Derrota}
    ECobra --> Troca[Troca jogador]
    Troca --> FimJogo[Final do jogo: Pergunta "Jogar novamente?"]
    FimJogo --> INICIO
```
-
- ### “4. Exportando para .jpg”**
- Após criar o fluxograma, salve como **‘tumba_egipcia.jpg’**.
 - Se precisar, posso explicar como implementar uma parte específica (ex: sorteio aleatório).
- Quer que eu detalhe alguma parte? 🐍

Deepseek (GOUSMAS WAR)

PEDIDO :

crie um jogo no flowgorithm,Neste jogo de estratégia, os jogadores controlam criaturas chamadas

Gousmas, que podem se dividir e acumular fúria ao serem atacadas.

- Cada jogador inicia com duas Gousmas, cada uma com nível de fúria 1;

- Quando uma Gousma ataca outra, transfere todo o seu nível de fúria

para a Gousma atacada;

- Se uma Gousma atingir um nível de fúria maior que 5, ela se desintegra;

- O jogador pode optar por dividir uma Gousma, transferindo uma parte da

fúria para uma nova Gousma (respeitando o limite de duas Gousmas por

jogador);

- O jogador que perder todas as suas Gousmas será derrotado;

RESPOSTA DO DEEP SEEK :

Para criar um jogo no Flowgorithm que siga as regras descritas, você pode seguir os passos abaixo. O Flowgorithm é uma ferramenta visual para criar algoritmos, então você precisará desenhar o fluxograma com base nas regras do jogo. Vou descrever como você pode estruturar o algoritmo:

—

Passos para criar o jogo no Flowgorithm:

1. **Inicialização do Jogo:**

- Declare variáveis para representar as Gousmas de cada jogador (por exemplo, 'GousmaJogador1_1', 'GousmaJogador1_2', 'GousmaJogador2_1', 'GousmaJogador2_2').
- Inicialize o nível de fúria de cada Gousma com 1.

2. **Loop Principal do Jogo:**

- Crie um loop que continua enquanto ambos os jogadores tiverem pelo menos uma Gousma.
- Dentro do loop, exiba o estado atual das Gousmas de cada jogador (nível de fúria).

3. **Menu de Ações:**

- Para cada jogador, ofereça as opções:
 - **Atacar:** O jogador escolhe uma de suas Gousmas para atacar uma Gousma do oponente.
 - **Dividir:** O jogador escolhe uma de suas Gousmas para dividir, transferindo parte da fúria para uma nova Gousma (se ainda não tiver duas Gousmas).
- **Passar a vez:** O jogador não faz nada.

4. **Lógica de Ataque:**

- Se o jogador escolher atacar, transfira todo o nível de fúria da Gousma atacante para a Gousma atacada.
- Verifique se a Gousma atacada atingiu um nível de fúria maior que 5. Se sim, ela se desintegra (remova-a do jogo).

5. **Lógica de Divisão:**

- Se o jogador escolher dividir, transfira metade (ou outra quantidade) da fúria da Gousma selecionada para uma nova Gousma.
- Certifique-se de que o jogador não exceda o limite de duas Gousmas.

6. **Verificação de Fim de Jogo:**

- Após cada ação, verifique se algum jogador perdeu todas as suas Gousmas.
- Se um jogador perder todas as Gousmas, declare o outro jogador como vencedor e encerre o jogo.

—

Exemplo de Fluxograma:

1. **Início:**

- Inicialize as variáveis das Gousmas.

2. **Loop Principal:**

- Exiba o estado das Gousmas.
- Pergunte ao jogador 1 qual ação deseja realizar.
- Execute a ação escolhida.
 - Verifique se o jogador 2 perdeu todas as Gousmas.
- Repita o processo para o jogador 2.

3. **Ação de Ataque:**

- Transfira a fúria.
- Verifique se a Gousma atacada se desintegrou.

4. **Ação de Divisão:**

- Transfira parte da fúria para uma nova Gousma.
- Verifique se o limite de Gousmas foi respeitado.

5. **Fim de Jogo:**

- Declare o vencedor.

—

Exemplo de Código (pseudocódigo):

```
'''plaintext
```

```
Início
```

```
    // Inicialização
```

```
    GousmaJogador1_1 = 1
```

```
    GousmaJogador1_2 = 1
```

```
    GousmaJogador2_1 = 1
```

```
    GousmaJogador2_2 = 1
```

```
    // Loop do jogo
```

```
    Enquanto (Jogador1 tem Gousmas E Jogador2 tem Gousmas) faça:
```

```
        // Turno do Jogador 1
```

```
        Exibir estado das Gousmas
```

```
        Ação = Escolher ação (Atacar, Dividir, Passar)
```

```
        Se Ação == "Atacar":
```

```
            Executar ataque (Jogador1, Jogador2)
```

```
        Senão Se Ação == "Dividir":
```

```
            Executar divisão (Jogador1)
```

```
        Fim Se
```

```
        // Verificar se Jogador 2 perdeu
```

```
        Se Jogador2 não tem Gousmas:
```

```
            Exibir "Jogador 1 venceu!"
```

```
        Fim
```

```
        Fim Se
```

```
        // Turno do Jogador 2
```

```
        Exibir estado das Gousmas
```

```
        Ação = Escolher ação (Atacar, Dividir, Passar)
```

```
        Se Ação == "Atacar":
```

```
            Executar ataque (Jogador2, Jogador1)
```

```
        Senão Se Ação == "Dividir":
```

```
            Executar divisão (Jogador2)
```

```
        Fim Se
```

```
        // Verificar se Jogador 1 perdeu
```

```
        Se Jogador1 não tem Gousmas:
```

```
            Exibir "Jogador 2 venceu!"
```

```
        Fim
```

```
        Fim Se
```

```
    Fim Enquanto
```

```
Fim
```

```
'''
```

```
—
```

Observações:

- No Flowgorithm, você precisará usar formas como "Processo", "Decisão", "Entrada/Saída" e "Loop" para representar cada parte do algoritmo.
- Teste o jogo para garantir que todas as regras sejam seguidas corretamente.

Se precisar de mais ajuda ou de um exemplo mais detalhado, é só avisar! 😊