Relatório do Jogo de Adivinhação

Sumário

Relatório do Jogo de Adivinhação 1
1. Fluxo do Jogo 1
2. Construção do Código 1
a. Estrutura Básica 1
b. Variáveis Utilizadas 1
c. Laço de Jogo 2
d. Decisões de construção 2
3. Decisões de Design 3
4. Conclusão 3

1. Fluxo do Jogo

O jogo de adivinhação começa com uma saudação ao jogador, explicando as regras e instruções básicas: o jogador deve adivinhar um número secreto gerado aleatoriamente entre 1 e 20. Durante o jogo, o jogador pode digitar "sair" a qualquer momento para encerrar a partida. O jogo oferece feedback a cada tentativa, indicando se o número secreto é maior ou menor que a tentativa do jogador. Quando o jogador adivinha o número correto, o jogo informa quantas tentativas foram necessárias. Ao final de cada partida, o jogador é convidado a jogar novamente ou encerrar o jogo.

2. Construção do Código

a. Estrutura Básica

O código utiliza um loop do-while para garantir que o jogador tenha pelo menos uma tentativa de adivinhar o número. O loop principal do jogo envolve:

- A geração de um número secreto aleatório.
- A captura das tentativas do jogador.
- A verificação se o jogador deseja sair.
- A comparação entre a tentativa e o número secreto.

Caso o jogador não tenha acertado, o jogo continua dando dicas até que ele acerte ou decida encerrar.

b. Variáveis Utilizadas

- **jogarNovamente**: Armazena a resposta do jogador para decidir se deseja iniciar uma nova partida após o término da anterior.
- **numAleatorio**: Gera um número aleatório entre 1 e 20 para ser o número secreto do jogo.

- **numSecreto**: Armazena o número secreto gerado para ser adivinhado pelo jogador.
- **tentativas**: Um contador que mantém o número de tentativas realizadas pelo jogador.
- **tent**: Armazena cada tentativa que o jogador faz (o número inserido).
- **entrada**: Armazena o valor digitado pelo jogador antes de ser validado e convertido em número.

c. Laço de Jogo

O jogo utiliza um laço do-while para a execução contínua até que o jogador adivinhe corretamente ou opte por sair. No laço interno:

- O jogador é solicitado a inserir uma tentativa.
- O valor digitado é verificado para determinar se é "sair" ou um número válido entre 1 e 20.
- Se for "sair", o laço é interrompido, e o jogador sai da partida.
- Se for um número válido, ele é comparado com o número secreto. Dependendo da comparação, o jogo fornece dicas se o número é maior ou menor.
- Quando o número correto é adivinhado, o jogo exibe o número de tentativas que o jogador fez até acertar.

d. Decisões de construção

1. **Reutilização e organização das variáveis**: A variável tent armazena o valor da tentativa do jogador, enquanto tentativas conta o número de vezes que o jogador tentou adivinhar. Isso mantém o controle claro das tentativas.
2. **Tratamento de Erros**: O código verifica se o valor inserido é válido usando int.TryParse(). Se o valor não for um número, o jogo exibe uma mensagem de erro solicitando uma nova tentativa. Isso evita que o jogo falhe com entradas inesperadas.
3. **Limitação de valores**: O código verifica se a tentativa do jogador está dentro do intervalo esperado (1 a 20). Se o jogador digitar um número fora desse intervalo, o jogo exibe uma mensagem explicando que o valor é inválido, o que garante que as regras do jogo sejam seguidas.
4. **Flexibilidade de Saída**: A palavra "sair" pode ser digitada a qualquer momento, encerrando o jogo de forma limpa. Isso oferece ao jogador uma maneira simples de terminar a partida sem precisar adivinhar o número até o fim.
5. **Pluralidade nas mensagens**: O código utiliza uma verificação simples para ajustar a mensagem final, de modo que a palavra "tentativas" apareça no plural ou singular, dependendo se o jogador acertou na primeira tentativa ou não. Isso melhora a experiência do usuário com uma interface de linha de comando mais natural e amigável.
6. **Jogabilidade Contínua**: Após o término de cada partida, o jogo pergunta se o jogador deseja jogar novamente. Esse ciclo de repetição permite uma jogabilidade contínua até que o jogador escolha encerrar o jogo.

3. Decisões de Design

A construção do código foi feita de forma a garantir uma experiência de jogo simples, eficiente e amigável. Algumas decisões de design principais incluem:

- **Facilidade de Uso**: O jogo foi projetado para ser simples de usar, com instruções claras exibidas logo no início.
- **Tratamento de Entradas Inválidas**: Implementar o tratamento de entradas inválidas foi uma decisão crucial para evitar travamentos ou erros durante a execução.
- **Design Modular e Reutilizável**: As funções e lógicas do jogo estão organizadas de maneira a serem facilmente reutilizadas. O laço do-while é estruturado para facilitar o fluxo contínuo do jogo, permitindo que o jogador jogue várias partidas consecutivas sem necessidade de reiniciar o programa.
- **Ciclos de Jogo e Feedback**: O jogador recebe feedback imediato após cada tentativa, o que o mantém engajado e direciona sua próxima tentativa. O jogo informa se o número secreto é maior ou menor e exibe uma mensagem final quando o jogador acerta.

4. Conclusão

O jogo de adivinhação foi projetado para oferecer uma experiência interativa e intuitiva, com um fluxo de jogo contínuo e feedback instantâneo para o jogador. O tratamento de entradas inválidas e a possibilidade de sair a qualquer momento tornam o programa fácil de usar e flexível. A estrutura de repetição e a separação clara das variáveis facilitam a manutenção e escalabilidade do código. O design focou na simplicidade e na usabilidade, com decisões estratégicas para melhorar a experiência do jogador e manter o código eficiente.