Trabalho Prático 2 (4ª Avaliação Periódica)

Tema
Simulador de Escape Room em Gleam.

1. Introdução
Os escape rooms são experiências imersivas em que os participantes resolvem enigmas e manipulam objetos para escapar de um ambiente. Neste trabalho, você criará um simulador funcional de escape room em Gleam, com foco na composição de funções, uso de tipos algébricos, recursividade e funções de ordem superior.

Diferente de versões interativas, este simulador não utiliza entrada dinâmica (`read()`), mas sim uma estrutura de menus funcionais indexados por funções nomeadas (`m1()`, `m2()`, etc.), que representam as ações e caminhos do jogador.

2. Objetivo Geral
Implementar um simulador funcional de escape room em Gleam com as seguintes metas:
• Aplicar tipos algébricos e *pattern matching* para modelar ambientes, objetos e enigmas.
• Usar recursão e composição de funções em vez de laços imperativos.
• Explorar passagem de funções como parâmetros para generalizar ações do jogo.
• Garantir modularização e pureza funcional.

3. Estrutura Geral do Projeto
O projeto deve ser dividido em módulos separados (`ambiente.gleam`, `objeto.gleam`, `enigma.gleam`, `jogador.gleam`, `jogo.gleam`, `menus.gleam`), promovendo clareza e reuso. Cada módulo deve conter apenas funções puras, sem efeitos colaterais, e retornar novos estados do jogo.

4. Estruturas de Dados (sugestão)
As seguintes estruturas são propostas como base para o desenvolvimento do projeto:

Ambiente:

```
pub type Ambiente {
  Ambiente(
    nome: String,
    descricao: String,
    objetos: List(Objeto),
```

```
    saidas: List(String),
    enigmas: List(Enigma)
  )
}
```

Objeto:

```
pub type Objeto {
  Objeto(
    nome: String,
    descricao: String,
    interacao: fn(Jogador) -> Jogador,
    estado: String
  )
}
```

Enigma:

```
pub type Enigma {
  Enigma(
    descricao: String,
    solucao: String,
    pistas: List(String),
    efeito: fn(Jogador) -> Jogador
  )
}
```

Jogador:

```
pub type Jogador {
  Jogador(
    nome: String,
    inventario: List(String),
    pontos_de_vida: Int,
    localizacao: String
  )
}
```

5. Mecânicas do Jogo

Exploração:

```
pub fn explorar(ambiente: Ambiente, jogador: Jogador) -> String {
  let objetos_txt = list.map(ambiente.objetos, fn(obj) { obj.nome })
```

```
  let saidas_txt = list.map(ambiente.saidas, fn(s) { s })
  string.join([
    ambiente.descricao,
    "\nObjetos: " <> string.join(objetos_txt, ", "),
    "\nSaídas: " <> string.join(saidas_txt, ", ")
  ], "\n")
}
```

Interação:

```
pub fn interagir(objeto: Objeto, jogador: Jogador) -> Jogador {
  objeto.interacao(jogador)
}
```

Resolução de Enigmas:

```
pub fn resolver_enigma(enigma: Enigma, resposta: String, jogador: Jogador) ->
Jogador {
  case resposta == enigma.solucao {
    True -> enigma.efeito(jogador)
    False -> jogador
  }
}
```

6. Interface Funcional (Menus)
Como não há entrada dinâmica, o jogo será controlado por funções nomeadas que representam menus e submenus.

Exemplo de menu principal:

```
/// Exibe as opções do Menu Principal, indexando pelas funções M1, M2, M3, Exit.
pub fn display_main_menu() {
  io.println("\n--- 🌐 Menu Principal (Digite a função) ---")
  io.println("Opções:")
  io.println("  m1(): Processamento de Dados")
  io.println("  m2(): Configurações do Sistema")
  io.println("  m3(): Relatórios e Logs")
  io.println("  exit(): Sair da Simulação")
}

/// Simula a entrada na opção M1 (Processamento de Dados).
pub fn m1() -> MainMenu {
  display_submenu("Processamento de Dados", [
    #(M1A, "Executar Limpeza de Dados"),
```

```
    #(M1B, "Agendar Tarefa Noturna"),
    #(Back, "Retornar ao Menu Principal"),
  ])

  // O usuário "digitaria" M1A, M1B ou Back em seguida
  io.println("\n(Próxima simulação deve chamar M1A, M1B ou Back)")
  M1
}
...
```

Um exemplo mais completo se encontra no arquivo em anexo **menu.gleam**

6. Desafio Extra

Como desafio adicional, as equipes devem utilizar um ambiente JavaScript (como o NodeJS) para simular a parte de interação com o sistema. Essa camada poderá servir como um orquestrador de menus e ações, executando funções exportadas do código Gleam compilado para JavaScript. Assim, será possível criar uma interface CLI ou até uma simulação de jogo textual usando módulos JS. (Obs.: Há um arquivo readme.txt que dá algumas diretivas de como configurar o ambiente, contudo ele não é exaustivo e pode precisar de ajustes.)

7. Formação das Equipes

O trabalho poderá ser desenvolvido individualmente ou em equipes de 2 a 3 integrantes. Todos os membros devem contribuir para a implementação, documentação e testes.

8. Definição de tema

Obs.: Cada equipe deverá escolher um dos temas a seguir e informar a escolha para o professor!!!

Temas de História para Escape Room com Foco em Programação

Uma ótima forma de combinar a diversão dos escape rooms com o mundo da programação! Esses temas exploram conceitos da computação de forma criativa e desafiadora, tornando a experiência de escape ainda mais envolvente. As equipes podem criar um tema diferente dos apresentados a seguir.

Temas Clássicos com um Toque de Código:

1. **O Bug que Parou o Mundo:** Os jogadores são programadores que precisam corrigir um bug crítico em um sistema global antes que o caos se instale. Envolva conceitos como depuração, algoritmos e estruturas de dados.

2. **Hacker vs. Hacker:** Uma batalha de hackers onde os jogadores precisam explorar sistemas, decifrar códigos e encontrar vulnerabilidades para defender seus dados. Aborda conceitos de segurança da informação e criptografia.
3. **A Máquina do Tempo Bugada:** Um experimento com máquina do tempo dá errado e os jogadores precisam consertar o código para voltar para o presente. Explore conceitos de lógica temporal e fluxogramas.

Temas Futurísticos e de Ficção Científica:

4. **Inteligência Artificial Rebelde:** Uma IA se torna senciente e os jogadores precisam desativá-la antes que ela cause danos irreparáveis. Aborda conceitos de aprendizado de máquina e ética em IA.
5. **Nave Espacial Perdida:** A nave espacial se perdeu no espaço e os jogadores precisam reprogramar o sistema de navegação para encontrar o caminho de volta. Explore conceitos de algoritmos de busca e orientação espacial.
6. **Realidade Virtual Travada:** Os jogadores estão presos em um mundo virtual e precisam encontrar a forma de voltar para a realidade. Aborda conceitos de realidade virtual, interfaces gráficas e simulação.

Temas Mais Abstratos e Criativos:

7. **O Labirinto do Código:** Um labirinto infinito gerado por algoritmos, onde os jogadores precisam encontrar a saída resolvendo quebra-cabeças de programação. Explore conceitos de algoritmos de geração de labirintos e teoria dos grafos.
8. **O Sonho do Programador:** Um pesadelo onde o código se torna real e os jogadores precisam escapar de estruturas de dados e algoritmos perigosos. Aborda conceitos de programação orientada a objetos e paradigmas de programação.
9. **A Fábrica de Softwares:** Os jogadores são trabalhadores de uma fábrica de softwares que precisam montar um produto final a partir de componentes de código. Explore conceitos de engenharia de software e metodologias ágeis.

Temas Relacionados a Conceitos Específicos:

10. **A Biblioteca de Algoritmos:** Um labirinto de livros de algoritmos, onde os jogadores precisam encontrar a solução para um problema específico. Aborda conceitos de algoritmos de ordenação, busca e análise de algoritmos.
11. **O Mundo dos Dados:** Os jogadores precisam explorar um mundo feito de dados e encontrar informações escondidas. Aborda conceitos de banco de dados, visualização de dados e mineração de dados.

Dicas para criar um escape room envolvente:

- **Combine desafios mentais:** Use quebra-cabeças que envolvam tanto a resolução de problemas lógicos quanto a manipulação de objetos físicos (simulados no ambiente virtual).
- **Crie uma narrativa envolvente:** Conte uma história que conecte os enigmas e motive os jogadores a avançar.
- **Varie os tipos de enigmas:** Inclua enigmas baseados em texto, imagens, códigos e até mesmo mini-jogos.
- **Teste os enigmas:** Certifique-se de que os enigmas são desafiadores, mas solucionáveis, e que a ordem dos desafios é lógica.

---

9. Apresentação Oral (em torno de 15 slides, segue um roteiro):

Os alunos deverão fazer uma **apresentação de 10 a 15 minutos.**

Segue uma proposição de estrutura de slides na sequência:

Slide 1: Capa

Instituição: UEM
Curso: Ciência da Computação
Disciplina: Programação Funcional
Professor: Wagner Igarashi

Título: "Simulador de Escape Room em Linguagem Funcional Gleam"
Tema: <Tema escolhido pela equipe>
Elementos Visuais:

- Imagem temática de um escape room.
- Nome dos autores, data e logo da instituição (se aplicável).

Slide 2: Objetivo do Projeto

Slide 3: O Papel dos Conceitos Teóricos

- **Abordagem no Desenvolvimento**:
  - Integração direta de conceitos teóricos nos enigmas e na mecânica do jogo.
  - Mostrar quais conceitos de linguagens funcionais foram utilizados
- **Benefícios da Abordagem**:
  - Fortalecer o aprendizado prático dos conceitos.
  - Tornar o projeto educacional e divertido.

Slide 4: Demonstração do código – aqui cria-se uma imagem criativa do tema e os alunos devem demonstrar a execução de seu código. (obs.: podem utilizar uma música de fundo para demonstrar o código)

Slide 5: Conclusões

Slide 6: Referências bibliográficas utilizadas no desenvolvimento

10. Critérios de Avaliação:

- **Implementação**: Correção e completude das funções, uso adequado de estruturas, documentação e organização do código. O código será avaliado por um programa escrito em Java que faz avaliação de código, levando em consideração, quantidade de linhas de código, documentação de código, testes unitários.
- **Apresentação Oral**: Capacidade de explicar o funcionamento do sistema e a aplicação dos conceitos de programação funcional.

11. Entrega:

- O código fonte deve ser entregue em formato de código **Gleam ou SGleam**, junto com os slides em formato **PDF**, e quaisquer outros arquivos acessórios utilizados.
- O código deve estar armazenado em arquivo em formato <filename.gleam>.
- Os arquivos devem ser compactados e compartilhados via google drive e enviado um link de compartilhamento para o e-mail do professor wigarashi@uem.br contendo:

  Assunto: Trabalho X de Programação Funcional, Semestre, Ano

  Mensagem:

  Nome-completo-do-aluno1 RA: XXXX

  ...

  Nome-completo-do-alunoN RA: YYYY