Universidade Estadual de Londrina

Ciência da Computação – Estruturas de Dados I (2COP035)
Prof. Dr. Gilberto Fernandes Jr.

Prova/Trabalho - 1° Bimestre

Jogo de Exploração de *Dungeons*

1 O objetivo

O trabalho consiste em implementar um jogo baseado em exploração de *dungeons*, onde o jogador precisa encontrar a saída de um labirinto, **coletando itens**, **evitando armadilhas** e **enfrentando inimigos**.

A movimentação do jogador, as interações com o ambiente e os sistemas de combate utilizarão as estruturas de dados estudadas na disciplina: listas, pilhas e árvores.

2 Os requisitos:

A. Inventário do Jogador (Lista Encadeada)

- O jogador pode coletar itens ao longo da *dungeon*.
- Cada item coletado será inserido no final da lista.
    - Os itens podem ser desde poções para recuperar vida, repelentes para evitar encontrar inimigos por um número x de passos, ou itens que valem pontos a serem contabilizados no final.
- O jogo deve possuir uma opção de abrir o inventário com a listagem completa dos itens.
- Ao abrir o inventário o jogador poderá:
    - Usar um item que afete o personagem ou o ambiente (Fica à critério a quantidade e tipos de itens, porém, deve ser implementado **pelo menos dois tipos de item utilizável**)
    - Descartar um item, removendo-o de qualquer posição da lista.
    - Ver detalhes do item (escolher um item para exibir uma breve descrição)

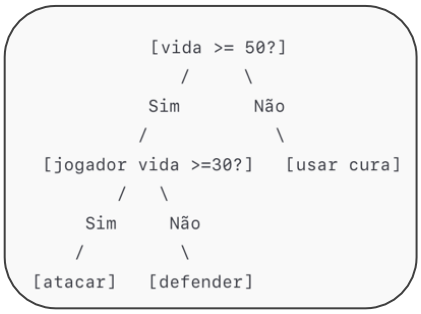
B. Sistema de Retrocesso de Movimento (Pilha)

- Implementar um sistema com pilhas para o jogador poder “desfazer” o último movimento realizado (cima, baixo, esquerda, direita)
- Na tela principal, deve haver uma opção de "voltar", a qual move o jogador de volta quantas vezes forem necessárias.

C. Sistema de Combate (Árvore Binária)

- O jogador pode enfrentar inimigos encontrados aleatoriamente no labirinto.
- Cada inimigo possui pontos de vida, um ou mais itens, e uma **Árvore Binária de Decisão** que orienta suas ações.

- A cada turno, a "IA" do inimigo avalia certas **condições** (vida dele, vida do jogador, etc.) e decide se **ataca**, **defende** ou **usa um item**.
- Exemplo:



- Já as ações do jogador são escolhidas pelo próprio usuário, conforme um menu com as opções de ataque, defesa, usar item e fugir.
- A fuga do personagem deve depender de algum sistema de controle, por exemplo, eficácia aleatória ou pode ter uma probabilidade menor de sucesso caso o inimigo esteja com a vida cheia.
- O combate ocorre executando alternadamente as ações de cada um até um dos dois ser derrotado.
- Implemente uma tela de combate a parte, e nela não é necessário que apareça o labirinto. Apenas as opções de combate (Atacar, defender, usar item e fugir). Mais detalhes abaixo.

D. Mapa da Masmorra:

- Representado por uma matriz NxN, com posições contendo:
  - Vazio — passagem livre.
  - Parede/obstáculo – passagem bloqueada
  - Item — coleta e adição ao inventário.
  - Armadilha — perda de pontos de vida.
  - Inimigo — inicia sistema de combate.
  - Saída — encerra o jogo com vitória.
- As posições dos itens, armadilhas e inimigos devem ser aleatórias.
- As armadilhas e inimigos ficam invisíveis aos usuários.
- O jogador começa sempre na mesma posição, definida como início da *dungeon*.

3 Funcionalidades Obrigatórias

- Sistema de movimentação e retrocesso via pilha.
- Sistema de coleta, remoção, listagem e utilização de itens via lista encadeada.
- Sistema de combate utilizando árvores binárias.
- Sistema de game over (por morte ou vitória).
- Estruturas de dados implementadas manualmente com TADs, sem o uso de bibliotecas.

- Implementar uma interface simples via terminal com menu para cada ação, com 3 tipos de exibição.
    - *Exibição padrão para movimentação*: labirinto + menu principal (voltar movimento, abrir inventário, desistir)
    - *Exibição de Inventário*: listagem de itens + menu (usar item, descartar item, detalhe do item)
    - *Exibição de Combate*: informações do player e do inimigo (pontos de vida) + menu (atacar, defender, usar item, fugir) + log de ações dos inimigos/usuários (escrever na tela cada ação realizada: inimigo curou, usuário atacou, inimigo atacou, etc.)
- EXTRA: se desejar, pode usar bibliotecas gráficas simples e com eventos de capturas de teclas, como a windows.h (nativa do Windows) ou ncurses (Mas o windows não possui ncurses nativo, então teria que usar Cygwin ou WSL), mas não é obrigatório. **Mas se fizer algo legal com interface, ganhará ponto extra na nota!**
- Criar arquivos .h e .c separados para cada estrutura: lista, pilha, árvore.
- Utilizar structs para representar jogador, item, inimigo.
- Modularizar funções: mover(), pegar_item(), batalhar(), etc. e se desejar, incluir em arquivos .h e .c para separar e organizar melhor.
- Verificar se haverá necessidade de usar funções de call-back (provavelmente sim!)

4 Entrega e Instruções Importantes

- Trabalhos inteiramente copiados da internet ou de demais colegas terão a nota **zerada**.
- A entrega será via *Google Classroom*.
- Entregar todo o projeto (com todos os arquivos necessários para sua execução) dentro de uma pasta zipada com o seguinte padrão de nomenclatura: P1_<Nome><Sobrenome>. Por exemplo: P1_GilbertoJunior
- O trabalho é INDIVIDUAL ou EM DUPLAS.
- O trabalho deverá ser apresentado somente a mim (não é necessário slides. Você irá apresentar o seu código, me explicar detalhadamente tudo o que fez e como as coisas funcionam, e responder perguntas que farei sobre o código).
- As datas de apresentação serão combinadas com a turma, e a ordem será por sorteio. As datas serão informadas claramente no Classroom.
- Qualquer dúvida, estou à disposição!

```
printf("Mãos à obra e bom trabalho!");
```