



Prof. Esther Colombini  
esther@ic.unicamp.br

## Atividade Prática 2: Herança, Relacionamentos e Polimorfismo

Entrega: 16/09/2023

### 1 Objetivo

O objetivo deste trabalho é permitir que o aluno aplique seus conhecimentos em herança, relacionamentos, e polimorfismo usando C++.

### 2 Passos da Atividade

#### Passo 1: Configuração do Ambiente

Antes de iniciar a atividade, certifique-se de que o ambiente de desenvolvimento C++ está configurado. Você pode usar qualquer IDE de sua escolha (Visual Studio Code, Code::Blocks, Dev-C++, etc.) ou até mesmo um editor de texto simples com o compilador g++ instalado. Lembre-se que a instalação do compilador g++ é dependente de plataforma.

#### Passo 2: Escrevendo o Código

Neste trabalho iremos desenvolver o restante do mini-game **Jewel Collector**. O objetivo deste jogo é que um robô, controlado pelo teclado, se desloque por um mapa 2D procurando comida e coletando as joias presentes no ambiente. Neste momento, iremos nos concentrar na implementação do robô e no relacionamento entre as classes para que o jogo funcione completamente. Para isso, você terá que implementar as seguintes classes:

1. **Map**: Modifique a classe Map para utilizar polimorfismo para armazenar os itens que aparecem no mapa. Altere a dimensão do mapa para  $30 \times 30$ .
2. **Robot**: A classe Robot é responsável por representar o robô no ambiente. O robô contém os atributos energy, e um **vector** que representa a sacola que o robô carrega consigo durante a exploração. O robô interage com o ambiente podendo usar os itens do mapa quando ele estiver em posições adjacentes a estes itens. O efeito do uso no robô depende das características do item: 1) os itens referentes a comida serão diretamente comidos para aumentar a energia; 2) as joias serão coletadas e guardadas na sacola para aumentar a pontuação do jogo; e 3) quando o robô encontra o tesouro no mapa, ele é capaz de coletar/comer todos os itens presentes no baú simultaneamente, aumentando de uma só vez a quantidade de energia e a pontuação do jogo. Após isso, o baú permanece vazio no ambiente e não é colocado na sacola. Lembre-se que o robô também é um item do jogo e deve ser implementado como herança da classe **item** criada anteriormente na Tarefa 3. Para interagir com o ambiente, o robô precisa conter um mapa em sua memória, lembre-se de associar o mapa ao robô para garantir que ele tenha as informações necessárias e atualizadas para navegar corretamente no ambiente.

Durante o jogo, o robô sempre perde 1 ponto de energia para cada passo que ele dá no ambiente. No início do jogo, ele começa com 10 pontos de energia. O robô é capaz de coletar ou comer apenas se as joias ou comidas estiverem em uma posição adjacente a ele. Lembre-se também de tratar os erros para os seguintes casos: 1) o robô tenta se deslocar para uma posição fora dos limites do mapa; 2) o robô tenta se deslocar para uma posição ocupada por outro item. Vale lembrar que joias, comidas e tesouros são intransponíveis. O robô é controlado pelo usuário, através dos seguintes comandos enviados pelo teclado: 1) **w** significa dar um passo ao norte, 2) **a** dar um passo a oeste, 3) **d** dar

um passo a leste, 4) **s** dar uma passo para ao sul, 5) **g** significa coletar/usar um item. Para cada comando executado pelo usuário, imprima o mapa, a energia e o score total da sacola do robô.

Para iniciar o jogo, popule o mapa com joias, comidas e tesouros em posições definidas por você. Demais escolhas ficam a critério do desenvolvedor. O jogo acaba quando o robô fica sem energia ou quando o robô coleta todas as joias do mapa.

### **Passo 3: Criando o Arquivo Makefile**

Um arquivo Makefile desempenha um papel fundamental na compilação de projetos maiores e mais complexos em linguagens de programação como C++ ou C. Ele automatiza o processo de compilação e torna a construção do projeto mais eficiente e organizada, permitindo o gerenciamento de dependências, a compilação incremental, a compatibilidade de plataforma e a documentação do processo de compilação.

Para o nosso projeto, crie um arquivo Makefile para compilar o programa `jogo.cpp` em um executável chamado **jogo**. Crie uma opção de apagar todos os `.obj` e todos os `.exe` criados.

### **Passo 4: Compilação e Execução**

Abra um terminal no diretório onde os arquivos `matriz.cpp` e `Makefile` estão localizados. Use os comandos `make` para compilar o programa e, em seguida, executar o executável gerado.