Documentação da Implementação: Jogo da Velha com Python e C

Nomes: Allan N. Schuch e João Linares

1. Apresentação da aplicação escolhida:

Para este trabalho, escolhemos implementar o clássico jogo da velha. A nossa ideia não era criar um jogo complexo, mas sim usar uma aplicação conhecida e com regras simples, de modo que o foco fosse fazer duas linguagens de programação com vocações bem diferentes, Python e C, trabalharem juntas para criar um programa funcional, que era o desafio do trabalho. O resultado foi um jogo da velha com uma interface gráfica simples onde dois jogadores podem competir, com toda a lógica de vitória e empate sendo gerenciada em segundo plano.

2. Qual linguagem implementa qual parte do código (Python para a interface, C para a lógica):

A arquitetura do nosso projeto foi pensada levando em consideração o que cada linguagem faz de melhor, sendo o Python usado para criar a interface do usuário de forma facilitada e o C para montar a lógica e fazer o jogo funcionar (nosso jogo não é complexo e não exige muito processamento, mas se fosse o caso, esta seria uma justificativa forte para se usar a linguagem C na lógica do jogo, para priorizar o desempenho). O projeto conta com dois arquivos, além do Makefile, cujo papel na implementação está descrito abaixo:

game.py (Python):

O Python foi a escolha natural para a parte visual da aplicação, pois conta com a biblioteca nativa Tkinter, que facilita bastante a criação de interfaces. O script game.py é responsável por criar toda a experiência do usuário, desde a janela principal do jogo até o tabuleiro com seus 9 botões clicáveis. Além disso, o código Python gerencia o fluxo do jogo do ponto de vista do usuário: ele captura os cliques do mouse, atualiza a tela exibindo os 'X' e 'O' nas posições corretas, informa de quem é a vez de jogar e, ao final, exibe as mensagens de vitória, derrota ou empate;

logic.c (C):

O C por sua vez foi usado para ser o "cérebro" do jogo. Todo o código em logic.c é focado puramente nas regras, sem se preocupar com a parte visual. Suas responsabilidades incluem manter o estado atual do tabuleiro em um array, validar se uma jogada é permitida (verificando se a posição está vazia) e checar após cada movimento se algum jogador completou uma linha, coluna ou diagonal para vencer. Ele também identifica quando não há mais jogadas possíveis e o jogo termina em empate. Essa separação garante que a lógica, que geralmente (não necessariamente nesse caso pois o jogo é bem simples) precisa ser rápida e eficiente, seja executada em C, enquanto a interface, que se beneficia de uma prototipagem mais ágil, fique a cargo do Python.

3. Método(s) utilizado(s) para permitir a interface entre as duas linguagens:

A parte mais interessante do projeto é como fizemos o código Python se comunicar com o código C. O método utilizado foi criar uma biblioteca compartilhada (ou shared library) a partir do código C e usar a biblioteca ctypes do Python para "conversar" com ela. O processo começa compilando o logic.c não como um programa normal, mas como um "pacote" de funções (logic.so em Linux/Mac ou logic.dll em Windows) que pode ser usado por outras aplicações. Nosso Makefile automatiza essa etapa. Em seguida, no game.py, a biblioteca ctypes do Python se encarrega de carregar esse arquivo na memória, tornando as funções C acessíveis.

Para que a comunicação funcione, precisamos estabelecer um "contrato", informando ao Python exatamente como as funções C se comportam: quais argumentos elas esperam (argtypes) e o que elas retornam (restype). Depois de fazer esse mapeamento, é possível chamar uma função C de dentro do Python de forma tão simples quanto chamar uma função nativa do Python. Por exemplo, quando o usuário clica em uma casa, o Python simplesmente chama a função makeMove da biblioteca C, passando a posição e o jogador. O código C faz toda a validação e retorna um valor indicando se a jogada foi bem sucedida.

Para garantir que está se mantendo a sincronia, o tabuleiro do jogo também é compartilhado. Usando ctypes, criamos uma variável em Python que recebe o mesmo valor do array gameBoard que existe na memória do código C. Dessa forma é criada uma conexão segura e eficiente entre as duas linguagens, permitindo que cada uma faça a sua função e se comunique com a outra, colaborando para o resultado final.