UFV Universidade Federal de Viçosa	Universidade Federal de Viçosa *Campus* Rio Paranaíba Sistemas de Informação SIN 211 - Estruturas de Dados Profª Rachel Reis

Projeto de Estruturas de Dados

"COMANDOS EM LOOP"

Introdução

O jogo Comandos em Loop tem como proposta aplicar os conceitos das estruturas de dados: pilha, fila e listas lineares. Basicamente o jogo consiste em um *player* (P), colocado em um tabuleiro, que deve alcançar um objetivo (O) desviando dos obstáculos (X) que encontrar no caminho. Para isso, serão apresentados um conjunto de comandos ao jogador, que deverá indicar a ordem e o número de vezes que cada comando deverá ser executado. O jogo encerra quando o jogador completar as três fases com sucesso.

Regras e Funcionamento do Jogo

1. O jogo deve ser jogado por uma pessoa e é formado por três fases. A tela inicial de cada fase deve inicialmente apresentar ao jogador:
   a. Um tabuleiro (Figura 1a).
   b. O sentido do *player*(P) no tabuleiro (Figura 1b).
   c. Um conjunto de comandos (Figura 1c).

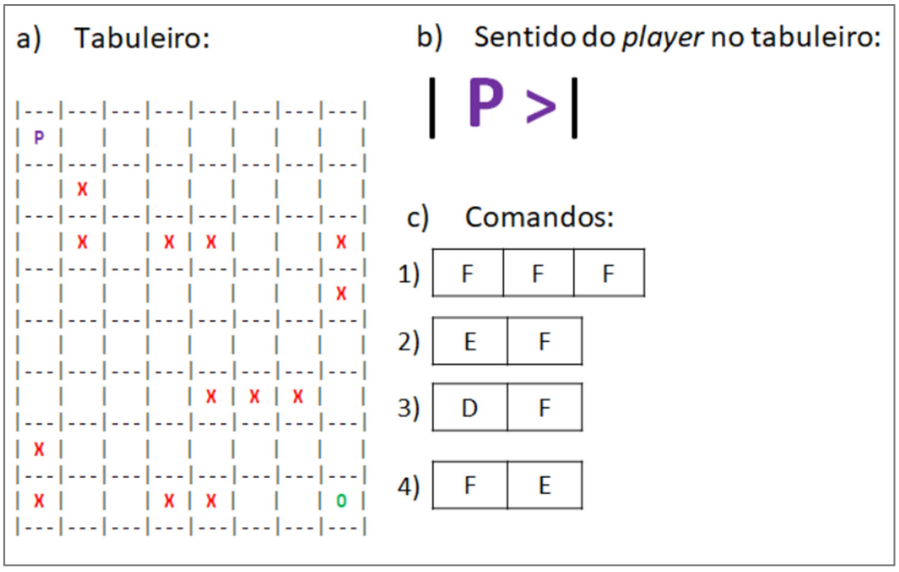

Figura 1 – Tela inicial do jogo.

2. Sobre o tabuleiro (Figura 1a):
    a. Cada tabuleiro deve ter dimensão 8x8.
    b. Ao iniciar o jogo em cada fase (1, 2 e 3), o *player* (P) deve sempre ser colocado na posição 0x0 e o objetivo (O) na posição 7x7.
    c. Cada tabuleiro deve possuir pelo menos seis obstáculos na horizontal e/ou vertical de tamanho 2 ou 3. Os mesmos devem ser espalhados no tabuleiro, ou seja, não devem se concentrar em um ponto.
    d. Os tabuleiros das três fases (1, 2 e 3) podem ser iguais ou não, mas para cada fase um conjunto diferente de comandos (ver Seção 4) deverá ser definido pelo programador.

3. Sobre o sentido do *player* (P) no tabuleiro:
    a. O *player* pode assumir os seguintes **sentidos** no tabuleiro:
        i. Para direita (>)
        ii. Para esquerda (<)
        iii. Para cima (^)
        iv. Para baixo (v)

4. Sobre os comandos:
    a. O programador deverá definir um conjunto diferente de **comandos** (no mínimo três e no máximo quatro) para cada fase do jogo (1, 2 e 3).
    b. Cada comando é formado por no mínimo 2 e no máximo 4 ações.
    c. O jogo permite as seguintes ações do *player* no jogo:
        i. **F**: avançar uma casa no sentido apontado pelo *player*. Se durante a movimentação, a borda do tabuleiro ou um obstáculo (X) for encontrado na próxima posição, o *player* deve permanecer na posição anterior.
        ii. **E**: rotacionar o sentido do *player* uma posição para esquerda sem movimentá-lo no tabuleiro. Por exemplo, se o sentido do player estiver para direita (>), a ação E fará com que o sentido mude para cima (^).
        iii. **D**: rotacionar o sentido do *player* uma posição para direita sem movimentá-lo no tabuleiro. Por exemplo, se o sentido do player estiver para direita (>), a ação D fará com que o sentido mude para baixo (v).
5. Fases do jogo:
    a. O jogo é formado por três fases.
        i. **Fase 1**: o *player* (P) deve sair da posição 0x0 e alcançar o objetivo (O) na posição 7x7. A sequência de comandos deve ser indicada pelo jogador (dentre aquelas mostradas na tela inicial) e armazenada na estrutura de dados **fila dinâmica**. Cada elemento da fila é formado pelo número do comando e o número de vezes que mesmo será executado.
        ii. **Fase 2**: o *player* (P) deve sair da posição 0x0 e alcançar o objetivo (O) na posição 7x7. A sequência de comandos deve ser indicada pelo jogador (dentre aquelas mostradas na tela inicial) e armazenada na estrutura de dados **pilha dinâmica**. Cada elemento da pilha é formado pelo número do comando e o número de vezes que o mesmo será executado.

iii. **Fase 3**: o *player* (P) deve sair da posição 0x0, alcançar o objetivo (O) na posição 7x7 e retornar para a posição inicial 0x0. A sequência de comandos de **ida** deve ser indicada pelo jogador (dentre aquelas mostradas na tela inicial) e armazenada na estrutura de dados **fila dinâmica**. Por outro lado, a sequência de comandos de **volta**, também indicada pelo jogador, deverá ser armazenada na estrutura de dados **pilha dinâmica**. Cada elemento da fila e pilha é formado pelo número do comando e o número de vezes que o mesmo será executado.

6. Funcionamento do jogo:
    a. Mostrar a tela inicial (Figura 1) com o número da fase (1, 2 ou 3).
    b. Solicitar ao jogador que digite a sequência de comandos, e número de vezes que o comando será executado, para que o *player* (P) alcance o objetivo (O). A sequência deve ser armazenada na estrutura de dados indicada na Seção 5, de acordo com a fase.
    c. Executar a sequência de comandos armazenada na(s) estrutura(s) de dado(s). Cada vez que um comando for executado, o tabuleiro deve ser atualizado e mostrado na tela com a nova posição do *player* (P).
    d. Para finalizar a execução de uma fase, a estrutura de dados que armazena a sequência de comandos deve estar vazia.
    e. Quando o jogador alcançar o objetivo nas Fases 1 e 2 ou o ponto inicial (após ida e volta na Fase 3), o programa deve exibir uma mensagem de parabéns e passar para a próxima fase. Se o objetivo não for alcançado, a fase corrente deve ser repetida para que uma nova sequência de comandos seja fornecida pelo jogador.
    f. O jogo encerra nas seguintes situações:
        i. Perde o jogo: se o jogador errar três vezes a sequência de comandos de uma fase.
        ii. Ganha o jogo: se o jogador fornecer a sequência correta de comandos para alcançar o objetivo das Fases 1, 2 e 3.

Avaliação

A avaliação consiste na entrega do código do projeto, que será avaliado em 20 pontos, e de uma entrevista com o professor, avaliada em 10 pontos. Os alunos que zerarem a entrevista, automaticamente receberão nota **zero** no projeto.

Os alunos que não enviarem o código fonte e/ou não comparecerem à entrevista receberão automaticamente a nota **zero** tanto no projeto quanto na entrevista.

A implementação do projeto deve obrigatoriamente utilizar as estruturas de dados: pilha e fila. O uso de outras estruturas de dados adicionais é opcional. Além disso, serão avaliados:

a. Endentação do código e boas práticas de programação.
b. Verificação se o jogador ganhou ou não.
c. Correto funcionamento do jogo.
d. Plágio.
e. Entrevista com os membros da equipe.

Observações importantes

Grupos de no **MÁXIMO** 03 alunos da mesma turma. Serão aceitos grupos com alunos de turmas diferentes, somente se todos os membros se comprometerem a realizarem a entrevista no horário da turma da tarde (integral) ou da noite (noturno), conforme calendário de entrevista definido pelo professor. Todos os membros deverão comparecer a entrevista no mesmo dia e horário. Os alunos que não participarem da entrevista receberão nota zero.

Todos os grupos devem registrar o nome e matrícula dos membros na planilha do Google em uma das seguintes abas: membros do integral, membros do noturno ou membros do integral e noturno. O link da planilha encontra-se disponível no link abaixo e para editá-la é preciso estar logado no e-mail da UFV:
https://docs.google.com/spreadsheets/d/1VzOTP3Rmwq5AyIsv3MbAvjCjFmdCMxEhr_MgBJ5PiAo/edit?usp=sharing

O projeto deverá ser entregue em um arquivo compactado (ZIP, RAR, etc.) contendo o código fonte da solução. Nomeie o arquivo compactado da seguinte forma: Matricula1_Matricula2_Matricula3.

O projeto deverá ser entregue via PVAnet Moodle. Em nenhuma hipótese serão recebidos trabalhos por outro meio e fora do prazo.

Arquivos com problemas ou erros de compilação/execução são de inteira responsabilidade dos grupos. Os trabalhos que se enquadrarem em alguns desses casos receberão **nota zero**.

Os trabalhos que forem enviados sem o código fonte receberão **nota zero**.

A **entrega do projeto** no PVAnet deverá ser realizada até 20/03/2022 (domingo) às 23:59.

As **entrevistas** serão realizadas no período de 23/03/2022 a 01/04/2022.

Trabalhos copiados (ou extremamente semelhantes) receberão **conceito F** (fraude em avaliação).