Relatório de Linguagem de Programação I

Battleship Puzzles

Alunos :

Thyall D’greville Santiago de Azevedo

Vinicius Ribeiro

Objetivo

- O objetivo deste trabalho consiste na modelagem, simulação ou reprodução do jogo *Batleship* . Temos como ideia base a divisão do projeto em duas partes, a primeira para a criação de um número já pré definido de diferentes Puzzles e a segunda parte , o outro programa, terá uma maior interação com o usuário e seu objetivo será a leitura dos Puzzles tendo como possíveis parâmetros como tamanho do campo, nível de dificuldade e possibilidade de dicas, sendo esse último ainda posto em pauta pelo grupo. Mais a frente serão descritos os aspectos mais metodológicos e técnicos sobre implementação e justificativas.

Metodologia

→ Programa 1

- No primeiro programa a partir das entradas dada pelo o usuário como as dimensões vamos alocar um vetor de ponteiro para poder armazenar todos os dados do campo, ex: entrada = 12x10 o vetor teria tamanho de 12 vezes 10, entrada 10x10 o vetor teria tamanho 100 e assim por diante. Após isso iremos começar a colocar os barcos no vetor. Acreditamos que sendo apenas 1 vetor teríamos um pouco de maior dificuldade para as coordenadas, mas em contrapartida iremos ganhar em desempenho do que ter dois com mais de uma alocação, também para percorrer o vetor, acessar e colocar os "valores" dos barcos.

- Para colocar os barcos iremos utilizar da função *rand* para escolher o local da plotagem do barco variando de 0 até o tamanho do vetor-1, e iremos utilizar também da mesma função para saber se o barco estará na vertical ou horizontal variando de 0 a 1 (0 = vertical e 1 = horizontal). Após isso iremos colocar o barco na região escolhida, e todas as regiões vizinhas terão um valor alterado para sabermos que essa área não poderá receber outro barco.

Representação e padronização dos barcos

- Temos com predefinição a ordem para plotagem de barcos no vetor e também de qual será o próximo ponto após o primeiro ter sido decidido pela função *rand.*

Barcos: Nessa ordem e num total de 10 barcos
→ 1 de 4 posições
→ 2 de 3 posições
→ 3 de 2 posições
→ 4 de posições

Vertical/horizontal:
→ Sempre que for horizontal será da esquerda para direita
→ Sempre que for na vertical será de cima para baixo

Signos → Usaremos os símbolos de "<", ">", "^", "v", "o" e para agua "=".

Conclusão

→ Após a criação e plotagem dos Puzzles, eles serão armazenados em um arquivo para serem lidos pelo próximo programa.

→ Programa 2

- Baseado na Disposição dos barcos no Programa 1, o Programa 2 tem a função de mostrar um total ‘x’ elementos ao usuário para ele ter um ponto de partida para jogar o Battleship, esses elementos sendo partes aleatórias dos barco, como início, corpo ou fim do mesmo e essa totalidade de elementos sendo definida pelo usuário como medidor de dificuldade.

- Após o usuário decidir o nível dificuldade que jogará o puzzle o usuário poderá ver o mapa que será disposto nas Coordenadas que vão de A-O e 1-15. Devido a escolha de apenas um Vetor no Programa 1 as coordenadas A-O atuarão na base de Enum para saber qual índice acessar corretamente.
  Exemplo:
  Puzzle de dimensão 10x10.
  Entrada - > B, 5

  -> Como o Puzzle tem dimensões de 10x10 o Enum de cada letra vai equivaler a dezena daquela linha, A = 10, B = 20, C = 30 e assim por diante, somado de 5 que é a coluna daquele elemento subtraído 1 já que Vetor começa do Índice 0; Ou seja a **Entrada (B, 5)** equivale ao elemento do índice **24**.

- Para Facilitar iremos adicionar fator de “replay” ao Battleship Puzzle pensamos em colocar um histórico de jogadas que será guardado em um arquivo separado que ele poderá acessar as jogadas que ele fez e caso ele jogue uma coordenada repetida ele será notificado que já realizou aquela escolha e essa jogada não será computada, como sistema de registro também pensamos em utilizar um sistema de Lista de recorde que seria os usuários que terminou o jogo com menos jogadas com aquelas dimensões e a porcentagem de eficiência de suas jogadas (ideia ainda sendo planejada, a implementação no caso).

Conclusão

→ A segunda parte do projeto irá trabalhar diretamente com o usuário em relação a ato de jogar (número de partidas), visualização do campo e entrada de dados para configurações do campo como as dimensões.