Gabriel Fonseca (DRE: 120027146)

Paulo Cesar Silva (DRE: 118145548)

Wagner Palhano (DRE: 120056690)

CMT012: Introdução à Programação C/c++

Projeto Final - Relatório

JOGO “RESTA-UM”

Universidade Federal do Rio de Janeiro

– Junho 2021 –

ÍNDICE

INTRODUÇÃO

O propósito do trabalho é desenvolver o jogo “resta-um”, utilizando a linguagem de programação C, a fim de colocar em prática os conhecimentos teóricos adquiridos durante as aulas. Durante todo o processo foi utilizado o sistema operacional Windows.

1. SOBRE O JOGO

a. Descrição do jogo

O resta um é um jogo de raciocínio lógico para um único jogador no qual você joga em um tabuleiro cheio de buracos. Há peças para todos os buracos, exceto um. O objetivo é limpar o tabuleiro de todas as peças, exceto uma.

b. Regras do jogo

1. O objetivo do jogo é deixar apenas uma peça no tabuleiro. Sendo assim, o jogador deve escolher uma peça e fazê-la saltar sobre uma outra peça, removendo do tabuleiro a peça que foi “saltada”. Esse movimento deve ser feito sempre na horizontal ou na vertical.

2. O jogo termina quando é impossível fazer qualquer movimento no tabuleiro.

3. O jogador vence quando consegue deixar no tabuleiro apenas uma única peça.

2. SOBRE O CÓDIGO

a. Como o código foi construído.

A ideia principal envolvendo a codificação, foi a de imaginar o tabuleiro do jogo como uma matriz 7x7, porém, ao compararmos uma matriz quadrada com o tabuleiro do jogo, é possível notar que o tabuleiro do jogo tem 16 posições a menos do que uma matriz 7x7, que são as posições dos vértices da matriz mais as 3 posições em volta dos vértices. Para inutilizar essas posições, utilizamos a estrutura COORDENADA_BLOQ, que recebe as coordenadas bloqueadas na matriz codificada (inexistentes no tabuleiro do resta-um). Essas “coordenadas bloqueadas” são representadas pelo caracter espaço. Os “espaços vazios” (posições que não contém peças) são representados pelo caracter “." (ponto), e as posições que contém peças, são representadas pelo caracter “O”.

Construímos um código que auxilia o usuário em seu primeiro contato, deixando assim o mesmo mais confortável, tendo uma apresentação, indicando as regras e instruções que o jogador deve seguir.

Quando o usuário inicia o jogo, ele se depara com um tabuleiro com indicações de coordenadas, que será por onde ele deve se basear na escolha de uma peça e utilizando as setas do teclado, para mover a peça de lugar. Seguindo esses mesmos passos até o término do código.

Para cada jogada seguimos praticamente os mesmos passos no código, verificamos se a posição selecionada é válida, ou seja, possui uma peça, depois verificamos para qual direção o jogador deseja mover essa peça, sendo que, para isso, seguindo na direção selecionada, o tabuleiro precisa ter a configuração nessa exata ordem: peça selecionada -> peça que será “saltada” ou “comida” -> espaço vazio; Caso uma dessas condições não seja respeitada, a jogada é considerada inválida, e o jogador precisa iniciar sua jogada novamente, selecionando uma peça e depois uma direção.

O final do jogo, como anteriormente explanado nas regras, só acontece quando:

* o jogador desiste ou não há mais peças;

*o tabuleiro (matriz) contém apenas uma peça (caracter “O”);

Porém, devido a disponibilidade de tempo para desenvolvimento do projeto, definimos uma derrota apenas quando o jogador desiste (entra com a coordenada 0-0), pois dessa forma, quando o jogador não tiver mais jogadas para fazer, a única opção que irá restar para ele, é a de desistir.

Já a vitória, foi definida da seguinte forma: Como o jogador precisa deixar apenas uma peça no tabuleiro, e a configuração inicial do tabuleiro possui 32 peças, podemos presumir que ele precisa comer exatamente 31 peças. Portanto, utilizamos um contador de peças comidas, que conta quantas peças o jogador comeu, ou também, vendo por um outro ângulo, quantas jogadas válidas o usuário fez. Dessa forma, 31 é o número exato de jogadas válidas/peças comidas necessárias para a vitória.

Ao final do jogo, é exibida uma mensagem com o resultado final!

b. Funções utilizadas.

Em nosso código utilizamos 3 funções pertencentes a bibliotecas, sendo elas: system(“cls”), getch() e setlocale(LC_ALL, "Portuguese").

Além disso, implementamos outras 3:

1. inicio_de_jogo():

   Função que apresenta o jogo e mostra instruções ao usuário.

2. imprime_matriz():

   Função que imprime o tabuleiro e as coordenadas ao usuário.

3. resultado():

   Função que imprime o resultado final do tabuleiro ao usuário, junto de uma mensagem de vitória, ou derrota.

c. Estrutura implementada.

Em nosso projeto foi implementada a estrutura COORDENADA_BLOQ, que delimita na matriz os espaços que não possuem buracos no tabuleiro, isto é, espaços onde as peças não podem ser encaixadas. Com isso, foi possível criar a “cruz” utilizada para jogar, de forma computadorizada.

d. Bibliotecas extras (que não foram ensinadas no período).

1. conio.h.

   Utilizada para implementar a função “getch()”

2. locale.h.

   Utilizada apenas para conseguir implementar caracteres especiais na interface com a função setlocale(LC_ALL, "Portuguese");

3. DIFICULDADES NO DESENVOLVIMENTO DO TRABALHO

1. Disponibilidade horária dos usuários do grupo.
2. Dificuldades na implementação da biblioteca ncurses.h, levando ao uso da biblioteca conio.h.

   Durante o desenvolvimento, buscamos formas de deixar o jogo mais dinâmico e intuitivo para o usuário. Uma das ideias de implementação, foi a de utilizar as “setas direcionais” para o usuário escolher para onde deseja mover a sua peça, obviamente seguindo as regras do jogo. Para isso, inicialmente tentamos utilizar a biblioteca “ncurses.h”, que possui uma função que consegue “ler” uma tecla “seta direcional”. Porém, não conseguimos utilizá-la, e trocamos a mesma pela biblioteca “conio.h” que possui a função getch(), uma função que consegue “ler” uma tecla “seta”. Para podermos implementar a mesma, utilizamos uma informação do fórum “Clube do Hardware”, onde o usuário identificado como “vangodp”, demonstra como implementar a função! (Link na parte de referências).

3. Manipulação da matriz utilizada, de forma que copiasse o tabuleiro do jogo:

   Quando começamos a trabalhar no código, não sabíamos muito bem como manipular a matriz para representar o tabuleiro do jogo. Entretanto, chegamos a conclusão de que teríamos que “bloquear” 16 posições da matriz utilizada, fazendo

assim uma “matriz jogável”, ou seja, fazendo uma matriz no formato de cruz, como no tabuleiro original do resta um.

4. APRENDIZADOS NO DESENVOLVIMENTO DO TRABALHO

1. A utilização da função getch da biblioteca conio.h, para receber o input das teclas direcionais do teclado:

   Durante a implementação do nosso código, surgiu a necessidade de usar as “teclas direcionais” para que o jogo pudesse acontecer da maneira mais intuitiva possível. Sendo assim, acabamos aprendendo a fazer isso usando a função getch() da biblioteca conio.h, em conjunto com o comando “switch case”.

2. A utilização da estrutura para o recebimento de vetores.

I. Referências

1. Como capturar o valor das setas. Disponível em: https://www.clubedohardware.com.br/topic/1288306-como-capturar-o-valor-das-setas/
2. Acentuação de caracteres em C com locale.h. Disponível em: https://cursos.alura.com.br/forum/topico-acentuacao-de-caracteres-em-c-com-locale-h-38310
3. “peg solitaire”. Disponível em: https://www.youtube.com/watch?v=rVA6I3bAm-0 (Vídeo utilizado como referência para melhorar a interação com o jogador)
4. Regras do resta um. Disponível em: https://www.booktoy.com.br/resta-um-7645#:~:text=Escolha%20uma%20pe%C3%A7a%20para%20c%C3%A7ar,n%C3%A3o%20 for%20 poss%C3%ADvel%20fazer%20movimentos
5. Jogo resta 1 (solução simples). Disponível em: https://www.youtube.com/watch?v=X25-oOY_w1U

II. Sistema operacional utilizado

1. Windows 7

   Ambiente de desenvolvimento: Dev C++ e Windows PowerShell